COMMODORE
128
Felhasználói kézikönyv

NOVOTRADE

Commodore 128-as

FELHASZNÁLÓI KÉZIKÖNYV

Novotrade Rt., 1987

A könyv eredeti címe: Commodore 128 Personal Computer System Guide (1985)

Szerkesztő: ROCHLITZ ANDRÁS

A kiadásért felel RÉNYI GÁBOR, a NOVOTRADE Rt. igazgatója
Műszaki szerkesztő: Hajdú Árpád
ISBN 963 02 4871 9

Szedte a Nyomdaipari Fényszedő Üzem, Budapest

Készült a Sylvester János nyomdában, Szombathely
Felelős vezető: Hanuszek Béla igazgató

TARTALOMJEGYZÉK

I. FEJEZET BEVEZETÉS

1. alfejezet A kézikönyv használata ... 9
2. alfejezet A Commodore 128-as személyi számítógép áttekintése ... 13

II. FEJEZET A C128-AS ÜZEMMÓD HASZNÁLATA

3. alfejezet A BASIC programnyelv ... 19
4. alfejezet Haladó BASIC programozás ... 33
5. alfejezet Néhány, kizárólag C128-as üzemmódban használatos BASIC parancs és billentyűművelet ... 45
6. alfejezet A C128-asra jellemző szín-, animációs, és sprite-grafikus utasítások ... 55
7. alfejezet Hang és zene C128-as üzemmódban ... 73
8. alfejezet 80 oszlop használata ... 87

III. FEJEZET A C64-ES ÜZEMMÓD

9. alfejezet A billentyűzet használata C64-es üzemmódban ... 93
10. alfejezet Saját programok tárolása és használata C64-es üzemmódban ... 95

IV. FEJEZET A CP/M ÜZEMMÓD

11. alfejezet Bevezetés a CP/M 3.0 operációs rendszerbe ... 101
12. alfejezet File-ok, lemezek és meghajtók CP/M 3.0 módban ... 105
13. alfejezet A konzol és a nyomtató használata CP/M üzemmódban ... 111
14. alfejezet A legfontosabb CP/M 3.0 parancsok összefoglalása ... 115
15. alfejezet A Commodore 128-as további szolgáltatásai a CP/M 3.0-hoz 119

V. FEJEZET BASIC 7.0 KISLEXIKON

16. alfejezet Bevezetés ... 125
17. alfejezet BASIC parancsok és utasítások ... 129
18. alfejezet BASIC függvények ... 175
19. alfejezet Változók és operátorok ... 187
20. alfejezet Foglalt szavak és jelek ... 191

FÜGGELÉKEK

A) függelék	A BASIC nyelv hibaüzenetei	196
B) függelék	A DOS hibaüzenetei	199
C) függelék	Kapcsolók és csatlakozók a külső egységekhez	202
D) függelék	Képernyőkódok	205
E) függelék	Az ASCII és a CHR$ kódok	207
F) függelék	A képernyő és a színtár térképe	209
G) függelék	Számontartott matematikai függvények	211
H) függelék	Tártérkép	212
I) függelék	Vezérlő- és escape kódok	215
J) függelék	Gépi kódú monitor	218
K) függelék	A BASIC 7.0 rövidítései	224
L) függelék	A lemezparancsok összefoglalása	226
M) függelék	Szakkifejezések magyarázata	227

I. FEJEZET

BEVEZETÉS

1. ALFEJEZET

A kézikönyv használata

Mielőtt még továbblapozna ebben a kézikönyvben, olvassa el a számítógéphez mellékelt másik könyvet (Ismerkedés a Commodore 128-as személyi számítógéppel), amely segítséget nyújt az első lépések megtételéhez.
Ha saját programjai megírásához és futtatásához elsősorban a BASIC nyelv érdekli, olvassa el először az I. fejezet 2. alfejezetét, amely a Commodore 128-as három üzemmódját ismerteti. Utána olvassa el a II. fejezetet (A C128-as üzemmód használata), amelyből megismerheti a C128-as és a C64-es üzemmódokban használt BASIC programnyelvet, a Commodore 128-as billentyűzetét, néhány bonyolultabb parancsot, amelyeket egyaránt használhat C128-as és C64-es üzemmódban; megtudhatja, hogyan használjon egy sor új, a C128-as üzemmódra jellemző BASIC parancsot (szín, grafika és hang), valamint a 80 oszlopos képernyőlehetőséget.
Ha C64-es üzemmódban kívánja használni a BASIC nyelvet, olvassa el a III. fejezetet. Valamennyi Commodore 128 BASIC 2.0 parancsot használhatja C64-es üzemmódban. Ne feledje azonban, hogy a Commodore 128 BASIC 7.0 nyelv sokkal több BASIC parancsot tartalmaz, mint a BASIC 2.0, és hogy a C128 BASIC parancsok sokkal többet tudnak, és használatuk is könnyebb, mint a megfelelő BASIC 2.0 parancsoké. C64-es üzemmódban futtathatja a forgalomba kerülő sok ezer C64-es szoftver programcsomag bármelyikét.
Amennyiben CP/M üzemmódban kíván dolgozni, lapozza fel a IV. fejezetet. CP/M üzemmódban több ezer kereskedelmi forgalomban levő szoftver programcsomag közül választhat, beleértve a PERFECT sorozatot is (PERFECT WRITER, PERFECT CALC, PERFECT FILER), de természetesen Ön is írhat saját CP/M programokat.
Ha többet akar megtudni a BASIC 7.0 parancsokról, olvassa el az V. fejezetet, a BASIC 7.0 kislexikont. Ez részletesen tárgyalja a BASIC 7.0 parancsainak, utasításainak és függvényeinek használatát.
Ha mind az öt fejezetet végigolvasta és további információra van szüksége valamely Commodore 128-as témában, nézze meg a Függeléket, amely számos felvilágosítással szolgál, felsorolja az összes BASIC és DOS hibaüzenetet, valamint összefoglalja a lemezparancsokat. Az M) Függelékek egy kisszótár, amely a számítógépes szakkifejezések magyarázatát tartalmazza.

2. ALFEJEZET

A Commodore 128-as személyi számítógép áttekintése ... 14
C128-as üzemmód ... 14
C64-es üzemmód ... 14
CP/M üzemmód ... 14

A számítógép bekapcsolása ... 14

A szoftver használata ... 15

Üzemmódváltó táblázat ... 15

A Commodore 128-as személyi számítógép néhány tulajdonsága:

- kibővített BASIC nyelv (a Commodore BASIC 7.0) új parancsokkal és lehetőségekkel;
- 128 K RAM, amely opcionális RAM modulokkal 256 vagy 512 K-ra bővíthető;
- 40 és 80 oszlopos képernyő;
- hozzákapcsolható az új 1571-es gyors lemezmeghajtó egység;
- 2 MHz-es működés;
- CP/M 3.0 működési lehetőség;
- professzionális billentyűzet, egybeépített („nemzetközi 10-es") numerikus billentyűzettel;
- beépített gépi nyelv monitor.

A Commodore 128-as személyi számítógép tulajdonképpen három számítógép, mivel három elsődleges üzemmódban működik:

- C128-as üzemmód
- C64-es üzemmód
- CP/M üzemmód.

A következőkben röviden ismertetjük az egyes üzemmódok kínálta lehetőségeket:

C128-AS ÜZEMMÓD

C128-as módban a számítógép 128 K RAM-mal és kibővített BASIC nyelvvel (a BASIC 7.0-val) dolgozik. Több mint 140 parancsával, utasításával és függvényével a BASIC 7.0 pontosabban, könnyebben és jobban oldja meg az olyan bonyolult feladatokat, mint a grafikus ábrázolás, az animáció, a hang és a zene. C128-as üzemmódban választhatunk 40 és 80 oszlopos képernyő között és használhatjuk mind a 92 billentyűt. A billentyűzeten az Escape, a Tab, az Alpha Lock és a Help billentyűkön kívül numerikus billentyűzet is van. A beépített gépi nyelv monitor segítségével saját, gépi nyelven írt programokat is készíthetünk és nyomon követhetünk. Ezeket a programokat BASIC programhoz kapcsolva használhatjuk. C128-as üzemmódban használhatunk egy sor új Commodore perifériát, ilyen pl. az új gyors lemezmeghajtó egység, az egér és a 40/80 oszlopos kompozit video/RGBI monitor. Természetesen minden standard Commodore periféria csatlakoztatható az alapgéphez.

C64-ES ÜZEMMÓD

C64-es üzemmódban a Commodore 128-as éppen úgy működik, mint egy Commodore 64-es számítógép. A Commodore 128-as rendelkezik a kereskedelmileg sikeres Commodore 64-es minden tulajdonságával, így lehetővé teszi a számos, könnyen hozzáférhető C64-es szoftver használatát.

A Commodore 128-as tökéletesen kompatíbilis a standard Commodore 64-es perifériákkal: felhasználói csatlakozás (user port), soros eszközök, kazettás egység, joystick, kompozit videomonitor és tv-készülék. A gép C64-es üzemmódban BASIC 2.0 nyelven, 40 oszloppal és 64 K RAM-mal dolgozik. A főbillentyűzet a funkcióbillentyűk elhelyezésének kivételével ugyanaz, mint a Commodore 64-es számítógépen, és ugyanazok a grafikai, szín- és hanglehetőségek is.

CP/M ÜZEMMÓD

CP/M üzemmódban egy beépített Z80-as mikroprocesszor lehetővé teszi a Digital Research Incorporated által kidolgozott CP/M 3.0 változat használatát, és még egy sor új, a Commodore cég által kifejlesztett tulajdonsága van. CP/M üzemmódban, amelyet CP/M Plus-nak neveznek, a Commodore 128-as számítógép 128 K RAM-mal, 40/80 oszlopos képernyővel és teljes billentyűzettel (ezen belül a numerikus billentyűzettel, valamint további speciális billentyűkkel) működtethető. Csatlakoztatható a rendszerhez az új, gyors lemezmeghajtó egység (Commodore 1571), valamint a standard perifériák.
A II., a III. és a IV. fejezet (a 3-tól a 15. alfejezetig) részletesen ismerteti a sokoldalú Commodore 128-as személyi számítógép három üzemmódjának tulajdonságait és használatát.

A SZÁMÍTÓGÉP BEKAPCSOLÁSA

Mielőtt bekapcsolná a számítógépet, néhány dolgot a biztonság kedvéért ellenőriznie kell. Az első, hogy megnézi, hogy a billentyűzet tetején levő 40/80 billentyű a monitornak megfelelően van-e beállítva. Ha pl. 40 oszlopos a monitorja, a 40/80 billentyűnek felső állásban kell lennie. Ha a monitor 80 oszlopos, a 40/80 billentyűt le kell nyomni.
Ha a Commodore 1902-es dual monitort a 40 oszlopos formában használja, a 40/80 billentyűnek felső állásban kell lennie, a monitor elején levő csúszókapcsoló pedig legyen középen. 80 oszlopos képernyő esetén a 40/80 billentyűt le kell nyomni és a monitor elején található kapcsolót teljesen jobbra kell tolni.
Függetlenül attól, hogy melyik képernyőformátumot használja, ellenőrizze, hogy az ALL CAPS és a SHIFT LOCK billentyű felső állásban van-e. Ha nem, előfordulhat, hogy egyáltalán nem kap képet, vagy ismeretlen jeleket lát a képernyőn. (Lásd az 5. alfejezetet, ahol a C128-as üzemmódban használatos összes speciális billentyű leírása található.)

A SZOFTVER HASZNÁLATA

Ha a MAGIC VOICE beszédmodult használja, helyezze a modult a cartridge nyílásba, és miközben lenyomva tartja a C-billentyűt, kapcsolja be a gépet. *Soha ne tegyen cartridge-ot a gépbe bekapcsolt állapotban.*
Ha nehézségbe ütközne C64-es módban működtetnie a cartridge-ot, kikapcsolt gépbe helyezze be, tartsa lenyomva a C-billentyűt és kapcsolja be a gépet.
A Commodore 64-es géphez kapható külső CP/M 2.2 cartridge-ot ne helyezzen be a Commodore 128-asba, ui. a gépnek már van Z80-as mikroprocesszora a CP/M 3.0 számára. Ha a CP/M 2.2 cartridge-ot behelyezi, váratlan következményekkel járhat.
Ha olyan szoftvert használ, amelyhez fényceruza is szükséges, csatlakoztassa azt a gép jobb oldalán, a KI/BE kapcsoló mellett található 1-es vezérlőkapuba (Controller Port 1).

Megjegyzés: Ha Commodore 1902-es dual monitort használ, ne felejtse el a videokapcsolót a COMPOSITE vagy a SEPARATED állásról az RGBI állásba kapcsolni, amikor 40-ről 80 oszlopos képernyőre vált, s ugyanezt ellenkező irányba 80-ról 40 oszlopra való váltás esetén.

ÜZEMMÓDVÁLTÓ TÁBLÁZAT

Cél-üzem-mód	Az üzemmód, amiből ki akar lépni					
	Kikapcsolt gép	C128-as 40 oszlop	C128-as 80 oszlop	C64-es	CP/M 40 oszlop	CP/M 80 oszlop
C128-as 40 oszlop	1. 40/80 billentyű felső állásban 2. Kapcsolja be a gépet		1. Nyomja meg az ESC billentyűt, engedje el 2. Nyomja meg az X-et **vagy** 1. 40/80 billentyű felső állásban 2. Nyomja meg a RESET gombot	1. 40/80 billentyű felső állásban 2. Kapcsolja ki, majd be a gépet	1. 40/80 billentyű felső állásban 2. Kapcsolja ki, majd be a gépet	1. 40/80 billentyű felső állásban 2. Kapcsolja ki, majd be a gépet
C128-as 80 oszlop	1. Nyomja le a 40/80 billentyűt 2. Kapcsolja be a gépet	1. Nyomja meg, majd engedje vissza az ESC billentyűt 2. Nyomja le az X billentyűt **vagy** 1. Nyomja le a 40/80 billentyűt 2. Nyomja meg a RESET gombot		1. Nyomja le a 40/80 billentyűt 2. Kapcsolja ki, majd be a gépet	1. Nyomja le a 40/80 billentyűt 2. Vegye ki a CP/M rendszerlemezt a lemezegységből, ha szükséges 3. Kapcsolja ki, majd be a gépet	1. 40/80 billentyűt lenyomva 2. Vegye ki a CP/M rendszerlemezt a lemezegységből, ha szükséges 3. Kapcsolja ki, majd be a gépet
C64	1. Tartsa lenyomva a C-billentyűt 2. Kapcsolja be a gépet **vagy** 1. Helyezze be a C64-es cartridge-et 2. Kapcsolja be a gépet	1. Írja be a GO 64 parancsot, majd nyomja meg a RETURN billentyűt 2. a gép megkérdi: ARE YOU SURE? (Biztos?). Írja be Y (igen), nyomja meg a RETURN-t	1. Írja be GO 64, nyomja meg a RETURN billentyűt 2. A gép megkérdi: ARE YOU SURE? (Biztos?). Írja be: Y (igen), nyomja meg a RETURN-t		1. Kapcsolja ki a gépet. 2. 40/80 billentyű legyen felső állásban 3. Tartsa lenyomva a C-billentyűt, közben kapcsolja be a gépet **vagy** 1. Kapcsolja ki a gépet 2. Helyezze be a C64-es cartridge-ot 3. Kapcsolja be a gépet	1. Kapcsolja ki a gépet 2. A 40/80 billentyű legyen felső állásban 3. Tartsa lenyomva a C-billentyűt, közben kapcsolja be a gépet **vagy** 1. Kapcsolja ki a gépet 2. Helyezze be a C64-es cartridge-ot 3. Kapcsolja be a gépet

Cél-üzemmód	Az üzemmód, amiből ki akar lépni					
	Kikapcsolt gép	C128-as 40 oszlop	C128-as 80 oszlop	C64-es	CP/M 40 oszlop	CP/M 80 oszlop
CP/M 40 oszlop	1. Kapcsolja be a lemezegységet 2. Helyezze be a CP/M rendszerlemezt 3. A 40/80 billentyű legyen felső állásban 4. Kapcsolja be a gépet	1. Kapcsolja be a lemezegységet 2. Helyezze be a CP/M rendszerlemezt 3. A 40/80 billentyű legyen felső állásban 4. Írja be a BOOT parancsot 5. Nyomja meg a RETURN-t	1. Kapcsolja be a lemezegységet 2. Helyezze be a CP/M rendszerlemezt 3. A 40/80 billentyű legyen felső állásban Írja be a BOOT parancsot 4. Nyomja meg a RETURN-t	1. A 40/80 billentyű legyen felső állásban 2. Kapcsolja be a lemezegységet 3. Helyezze be a CP/M rendszerlemezt 4. Kapcsolja ki, majd be a gépet		1. Helyezze be a CP/M segédlemezt az egységbe 2. Ha megjelenik az A>, írja be DEVICECONOUT: = 40 COL 3. Nyomja meg a RETURN billentyűt
CP/M 80 oszlop	1. Kapcsolja be a lemezegységet 2. Helyezze be a CP/M rendszerlemezt 3. Nyomja le a 40/80 billentyűt 4. Kapcsolja be a gépet	1. Kapcsolja be a lemezegységet 2. Helyezze be a CP/M rendszerlemezt 3. Nyomja le a 40/80 billentyűt 4. Írja be a BOOT parancsot 5. Nyomja meg a RETURN billentyűt	1. Kapcsolja be a lemezegységet 2. Helyezze be a CP/M rendszerlemezt 3. A 40/80 billentyű legyen lenyomva 4. Írja be a BOOT parancsot 5. Nyomja meg a RETURN-t	1. Nyomja le a 40/80 billentyűt 2. Kapcsolja be a lemezegységet 3. Helyezze be a CP/M rendszerlemezt 4. Kapcsolja ki, majd be a gépet	1. Helyezze be a CP/M segédlemezt 2. Ha megjelenik az A>, írja be: DEVICECONOUT: = 80 COL 3. Nyomja meg a RETURN billentyűt	

II. FEJEZET

A C128-AS ÜZEMMÓD HASZNÁLATA

3. ALFEJEZET

A BASIC programnyelv

BEVEZETÉS

A direkt mód ... 20
A program mód ... 20

A billentyűzet használata
A billentyűzet karakterkészlete ... 20
A parancsbillentyűk használata ... 20
A funkcióbillentyűk ... 22
Grafikus karakterek megjelenítése a képernyőn ... 22
A BASIC nyelvű programok beírásának szabályai ... 22

Kezdjünk hozzá – A PRINT parancs
Számok kiírása ... 23
A kérdőjel mint a PRINT parancs rövid formája ... 23
Szöveg kiírása ... 23
Színek megjelenítése a képernyőn ... 24
A kurzorbillentyűk használata idézőjelben PRINT parancsnál ... 24

A programozás kezdete
Mi a program? ... 24
Sorszámok ... 24
A program megtekintése – a LIST parancs ... 25
Egyszerű ciklus – a GOTO utasítás ... 25
A számítógép tárának törlése – a NEW parancs ... 26
Színek használata a programban ... 26

A program szerkesztése
Egy sor törlése a programból ... 26
Egy sor másolása ... 26
Egy sor kicserélése ... 26
Egy sor megváltoztatása ... 26

Matematikai műveletek
Összeadás és kivonás ... 27
Szorzás és osztás ... 27
Hatványozás ... 27
A műveletek sorrendje ... 27
Zárójel használata a műveletek sorrendjének meghatározására ... 27

Állandók, változók és füzérek
Állandók ... 28
Változók ... 28
Füzérek ... 29

Mintaprogram

Programok tárolása és ismételt felhasználása
A lemez formálása – a HEADER parancs ... 30
Kimentés (SAVEing) lemezre ... 30
Kimentés kazettára ... 31
Betöltés (LOADing) lemezről ... 31
Betöltés kazettáról ... 31
Egyéb lemezzel kapcsolatos parancsok ... 31

BEVEZETÉS

A BASIC programnyelv különleges nyelv, amelynek segítségével kommunikálhatunk a Commodore 128-assal. A BASIC használata az egyik módja annak, hogy a géppel közöljük, mit kell tennie.
A BASIC-nek saját szókincse (amely *parancsokból, utasításokból és függvényekből* áll) és saját szerkezeti szabályai (szintaxis) vannak. A BASIC szókinccsel és szintaxissal egy sor instrukciót adhatunk a gépnek, ezt nevezzük *programnak*, amelyet a számítógép aztán végrehajt, azaz *futtat*.
BASIC nyelven kétféle módon lehet kommunikálni a Commodore 128-assal: a program keretén belül vagy azon kívül, *direkt* (közvetlen) *mód*on.

A direkt mód

A számítógép készen várja a BASIC utasításokat direkt módban, mihelyt bekapcsoljuk. Direkt módban le kell írni a parancsokat a billentyűkön és a RETURN billentyű lenyomásával kell őket bevinni a gépbe. A számítógép a direkt módban írt parancsokat a RETURN billentyű lenyomása után azonnal végrehajtja. A legtöbb BASIC parancsot mind direkt, mind pedig *program mód*ban lehet használni a Commodore 128-ason.

A program mód

Program módban egy sor speciális feladat megoldására viszünk be instrukciókat a gépbe. Minden instrukciót egymás után következő programsorokban adunk meg. A program egy utasítása 160 karakter hosszú lehet, ez a 40 oszlopos képernyő esetében négy teljes képernyősort jelent, míg 80 oszlop esetén két teljes képernyősort.
Ha beírta a programot, azonnal fel lehet használni a RUN parancs beírásával és a RETURN billentyű megnyomásával. A programot tárolni is lehet lemezen vagy szalagon, a DSAVE (vagy a SAVE) paranccsal, és visszahívni lemezről vagy szalagról a DLOAD (vagy a LOAD) paranccsal. Ez a parancs kimásolja a programot a lemezről vagy a szalagról, és elhelyezi a Commodore 128-as tárában. Ekkor a RUN paranccsal ismét lehet használni az illető programot. A későbbiekben majd még lesz szó ezekről a parancsokról. Az esetek többségében saját készítésű vagy készen vett programokkal fogja majd használni számítógépét. Direkt módban csak akkor dolgozunk, ha a LIST, a LOAD, a SAVE és a RUN parancsokkal szerkesztünk programot. Általában az a különbség a direkt és a program mód között, hogy a direkt módban írt parancsoknak nincs sorszámuk.

A BILLENTYŰZET HASZNÁLATA

A BASIC használata lényegében megegyezik C64-es és C128-as üzemmódban. A legtöbb billentyű és sok parancs, amelyekről a későbbiekben még szó lesz, mindkét módban használható BASIC programozáshoz.

A billentyűzet karakterkészlete

A Commodore 128-as billentyűzetén két különböző karakterkészlet van:

– nagybetűk és grafikus karakterek;
– nagy- és kisbetűk.

A 80 oszlopos képernyőformátum esetén egy időben mindkét karakterkészlettel dolgozhatunk. Így összességében 512 különböző karaktert lehet megjeleníteni a képernyőn. 40 oszlopos kijelzés esetén egyszerre csak egy karakterkészlet használható.

Ha 40 oszlopos képernyővel kapcsoljuk be a számítógépet, általában a nagybetűs/grafikus karakterkészletet használja a gép. Ez azt jelenti, hogy minden, amit beírunk, nagybetűvel jelenik meg. Ha a két készlet között váltani akar, nyomja meg a SHIFT és a C= (Commodore) billentyűt egyszerre. Gyakorlásképpen kapcsolja be a számítógépet és nyomjon le néhány betűt vagy grafikus karaktert. Utána nyomja meg a SHIFT és a C= billentyűt. Figyelje meg, hogyan váltakoznak a képernyőn a nagy- és a kisbetűk. Nyomja meg ismét a SHIFT és a C= billentyűt, ekkor visszatér a nagybetűs/grafikus karakterkészlethez.

A parancsbillentyűk használata

Parancsbillentyűknek azokat a billentyűket nevezzük, amelyek üzeneteket közvetítenek a számítógépnek. Némelyik parancsbillentyűt (mint pl. a RETURN-t) önmagában használjuk, másokat (pl. a SHIFT, a CTRL, a C= és a RESTORE) más billentyűkkel együtt. A következőkben bemutatjuk a parancsbillentyűk használatát:

RETURN Ha megnyomja a RETURN billentyűt, a beírtakat elküldi a számítógép tárába. A RETURN billentyű megnyomására a kurzor (a következő karakter helyét jelző kis villogó négyzet) a következő sorban jelenik meg.
Néha előfordul, hogy rosszul írja be a parancsot, vagy olyasvalamit ír be, amit a gép nem ért meg. Ekkor, a RETURN billentyű megnyomása után valószínűleg a következő üzenet jelenik meg a képernyőn: SYNTAX ERROR (szintaktikai hiba). Ezt hibaüzenetnek nevezzük.

Az A függelék felsorolja a hibaüzeneteket és a hiba kijavításának módját.
Megjegyzés: A könyvben megadott példákban a következő módon jelezzük, hogy meg kell nyomni a RETURN billentyűt:

RETURN

SHIFT A billentyűzet legalsó sorában két SHIFT (váltó) billentyű található. Hasonlóan a normál írógépbillentyűzethez, az egyik a jobb, másik a bal oldalon.
A SHIFT billentyűt háromféleképpen használhatjuk:
1. A kisbetűs/nagybetűs karakterkészletnél a SHIFT billentyűt ugyanúgy használjuk, mint az írógép váltóját. Ha a SHIFT billentyűt lenyomva tartjuk, nagybetűket ír a gép, vagy a kétkarakterű billentyűk felső karaktereit.
2. A SHIFT billentyűt speciális funkciók ellátására néhány más parancsbillentyűvel együtt is használhatjuk.
3. Ha a billentyűzet nagybetűs/grafikus készletre van beállítva, a SHIFT billentyűt bizonyos billentyűk homloklapján levő grafikus jelek vagy karakterek kiírására lehet használni. Bővebben l. a „Grafikus karakterek megjelenítése a képernyőn" c. részt.

SHIFT LOCK Ha ezt a billentyűt leütjük, zár, tehát a billentyű lent marad. Ilyenkor minden, amit beírunk, nagybetűvel lesz, ill. a kétkarakteres billentyű felső karaktere fog megjelenni. Ha nyitni akarjuk a zárat, nyomjuk meg ismét a SHIFT LOCK billentyűt.

A kurzor mozgatása C128-as módban úgy mozgatjuk a kurzort, hogy vagy a főbillentyűzet fölött jobbra található négy, nyíllal ellátott billentyűt használjuk, vagy a két CRSR billentyűt a főbillentyűzet legalsó sorának jobb oldalán.
A négy, nyíllal ellátott billentyű használata: C128-as módban egyszerűen úgy mozgatjuk a kurzort, hogy a kívánt irányba mutató nyíllal ellátott billentyűt lenyomjuk. (C64-es módban ezek a billentyűk nem használhatók.)
A CRSR billentyűk használata: A kurzor mozgatására mind C128-as, mind pedig C64-es módban használhatjuk a fő billentyűzet alsó sorának jobb szélén található két billentyűt:
- ha egymagában nyomjuk le a CR↕SR billentyűt, a kurzor *lefelé* megy;
- ha a CR↕SR és a SHIFT billentyűt együtt nyomjuk le, a kurzor *fölfelé* megy;
- ha egymagában nyomjuk le a C⇄RSR billentyűt, a kurzor *jobbra* megy;
- ha a SHIFT-tel együtt nyomjuk le a C⇄RSR billentyűt, a kurzor *balra* megy.

Ha több karakter távolságra kívánjuk a kurzort mozgatni, nem kell mindig újból lenyomni a billentyűt, csak lent kell tartani, és a kurzort addig mozgatni, amíg el nem érjük vele a kívánt helyzetet.
Figyelje meg, hogy amikor a kurzor elér a képernyő jobb szélére, „befordul", azaz újra megjelenik a következő sor elején. Ha balra halad, és eléri a képernyő szélét, az előző sor végére ugrik.
Ismerje meg alaposan a kurzorbillentyűket, mert a kurzor mozgatása nagyban megkönnyíti a programozást. Kis gyakorlással elérhető, hogy szinte gondolkodás nélkül, automatikusan mozgassa a kurzort.

INST/DEL Ez egy kettős funkciót ellátó billentyű. Az INST az INSerT (beszúrni), a DEL pedig a DELete (törölni) rövidítése.
Karakterek beszúrása: Ha egy sorba utólag karaktereket akar beszúrni, az INST/DEL billentyűt a SHIFT-tel együtt kell használnia. Tegyük fel, hogy kihagyott néhány karaktert a sorból, pl.:

KI L MENNEM

Először vigye vissza a kurzort a hibához:

KI ■ MENNEM

Ezután, míg lenyomva tartja a SHIFT billentyűt, nyomja meg az INST/DEL-t, amíg a hiányzó karakterekhez szükséges hely meg nem lesz:

KI ■ L MENNEM

Figyelje meg, hogy az INST nem mozgatja a kurzort, csak üres helyet biztosít a kurzor és a tőle jobbra levő karakter között. A javításnál egyszerűen írja be a hiányzó karaktereket:

KI KELL MENNEM

Karakterek törlése: Ha a DEL billentyűt lenyomja, a kurzor egy karakternyi helyet balra megy és törli az ott levő karaktert. Ez azt jelenti, hogy ha valamit törölni akar, vigye a kurzort az eltávolítani kívánt karakter jobb oldalára. Nézzük a példát:

PRINT "HIBLA"

Nem HIBLA-t, hanem HIBA-t akart írni. A szóvégi A-t megelőző helytelen L-et úgy kell törölni, hogy a kurzort az A-ra visszük, s ha megnyomjuk a DEL billentyűt, a kurzortól jobbra levő karakter (az A) automatikusan egy hellyel balra megy. Így megkapjuk a helyes sort:

PRINT "HIBA"

Az INST és a DEL együttes használata: Ezt a két funkciót együtt is lehet használni a hibás karakterek kijavítására. Először is vigye a kurzort a hibás karakterekre, és nyomja meg csak az INST/DEL billentyűt, hogy törölje a karaktereket. Utána nyomja meg a SHIFT billentyűt együtt az INST/DEL-lel, hogy a szükséges helyet megteremtse, majd írja be a javítást. Közvetlenül a törölni kívánt karakterekre is lehet írni, az INST-tel pedig új helyet csinálhatunk.

Control (CTRL) A Control billentyűt más billentyűkkel együtt speciális feladatok, ún. vezérlőfunkciók megoldására használjuk. Tartsa lenyomva a Control billentyűt, miközben lenyom egy másik billentyűt.
Készen kapható szoftvereknél, ilyen pl. a szövegfeldolgozó rendszer, gyakran használunk vezérlőfunkciókat.
Az egyik gyakran használt vezérlőfunkció a karakter és a kurzor színbeállítása. Ehhez tartsuk lenyomva a CTRL billentyűt, s közben nyomjunk meg 1 és 8 között egy tetszőleges számbillentyűt a billentyűzet legfelső sorából. Lehetőség van nyolc további szín kiválasztására is a C- billentyű segítségével, erre a későbbiekben visszatérünk.

RUN/STOP Ez a billentyű kettős funkciót lát el. Bizonyos feltételek mellett a RUN funkciót a SHIFT és a RUN/STOP billentyűk együttes lenyomásával használhatjuk. A billentyű STOP funkciójával lehet a program futása közben megállítani a programot vagy a kiírást. A legtöbb készen kapható programban azonban a RUN/STOP billentyű STOP funkciója szándékosan ki van iktatva, hogy a használó ne tudja megállítani a programot, mielőtt az a végére ér. Ha megállítaná, esetleg értékes adatok semmisülnének meg.

RESTORE A RESTORE billentyűt a RUN/STOP-pal együtt használjuk, hogy a számítógép visszatérjen alapállapotába. A legtöbb készen kapható programból kiiktatták a RESTORE billentyűt, ugyanazért, mint a RUN/STOP STOP funkcióját, hogy értékes adatok megsemmisítését megakadályozzák.

CLR/HOME A CLR a CLeaR (kitisztul, kiürül) rövidítése. A HOME a képernyő bal felső sarkára vonatkozik, ezt nevezzük Home helyzetnek (kiindulási helyzet). Ha ezt a billentyűt magában nyomjuk le, a kurzor visszatér a kiindulási helyzetbe. Ha a CLR/HOME-billentyűt együtt használjuk a SHIFT-tel, a képernyőről eltűnik minden és a kurzor visszamegy a HOME (kiindulási) helyzetbe.

Commodore billentyű (C=) Ennek a billentyűnek egy sor funkciója van, többek között a következők:

1. A C= billentyűvel lehet oda-vissza váltani a nagy-/kisbetűs (a billentyűk tetején levő betűk és karakterek), ill. a nagybetűs/grafikus karakterkészlet (a nagybetűk és a homloklapon levő grafikai szimbólumok) között. Ehhez egyidejűleg le kell nyomni a C= billentyűt és a SHIFT-et.

2. A billentyűvel lehet a kurzor színét kiválasztani a második nyolcas színskáláról. Tartsa lenyomva a C= billentyűt, és közben nyomjon le egy számbillentyűt 1-től 8-ig a felső sorból.
3. Ha a számítógép bekapcsolása közben lenyomva tartjuk a C= billentyűt, közvetlenül C64-es módba kerülünk.

A funkcióbillentyűk

A tízes numerikus billentyűkészlet fölött elhelyezkedő (fönt F1, F3, F5 és F7 jelű, homloklapján pedig F2, F4, F6 és F8 jelű) billentyűket *funkcióbillentyűk*nek nevezik. Ezeket programozni lehet C128-as és C64-es módban egyaránt (l. a II. fejezet 5. alfejezetben a billentyű parancs leírásokat, valamint az V. fejezetben a BASIC 7.0 kislexikont). Készen kapható szoftvereknél ezeket a billentyűket gyakran arra használják, hogy egyetlen billentyű lenyomásával lehessen valamilyen feladatot elvégezni.

Grafikus karakterek megjelenítése a képernyőn

Ha a billentyű homloklapjának jobb oldalán levő grafikus jelet akarjuk megjeleníteni, tartsuk lenyomva a SHIFT billentyűt, majd nyomjuk le a kívánt jelű billentyűt. A jobb oldali karaktereket csak akkor tudjuk kiíratni, ha a billentyűzet a nagybetűs/grafikus karakterkészletre van beállítva (ez a gép bekapcsolásakor az egyik lehetséges választék).

Ha a billentyű előlapjának bal oldalán levő grafikus karaktert kívánjuk megjeleníteni a képernyőn, tartsuk lenyomva a C= billentyűt, miközben lenyomjuk a kívánt másik billentyűt. A bal oldali grafikus karaktert bármelyik karakterkészletből lehet használni.

A BASIC nyelvű programok beírásának szabályai

A BASIC nyelv ismerete nélkül is lehet BASIC programokat beírni és használni. Igen gondosan kell azonban gépelni, mert egy géphiba miatt a számítógép visszautasíthatja az információt. A következő szempontok betartása a minimumra csökkentheti a programlista írásánál vagy másolásánál előforduló hibákat:

1. A szavak között nem döntő a szóköz használata, pl. a FORT = 1TO10 ugyanaz, mint a FOR T= 1 TO 10. A BASIC kulcsszavakat azonban ne szakítsuk meg szóközzel. (A BASIC kulcsszavak felsorolása az V. fejezet BASIC 7.0 kislexikonában található.)

2. Idézőjelbe bármilyen karaktert be lehet írni. Némelyik karakternek, ha idézőjelben használjuk, speciális funkciója van. Ezeket a könyv később részletesen tárgyalja.

3. Legyünk óvatosak az írásjelek használatával. A vesszőknek, kettőspontoknak és pontosvesszőknek külön, speciális funkcióik vannak, amelyeket ebben az alfejezetben még ismertetünk.

4. Számozott sorok befejezése után mindig nyomjuk le a RETURN billentyűt **(RETURN)**.

5. Soha ne írjunk 160 karakternél többet a program egy sorába. Ez 40 oszlopos képernyőnél négy teljes képernyősort jelent, 80 oszloposnál pedig két teljes képernyősort. Részletesebben l. a 8. alfejezetben.

6. Különböztessük meg világosan az l betűt és az 1-es számot, valamint az O betűt és a 0 számot.

7. A számítógép a REM-et követően egy programsorban semmiről nem vesz tudomást. A REM a REMark (megjegyzés) rövidítése. Ezt az utasítást arra használjuk, hogy bizonyos megjegyzéseket fűzzünk a programhoz, amelynek kilistázásakor bárki megtudhatja, mi történik az adott helyen.

Kövesse ezeket az irányelveket, amikor a következő példákat és programokat beírja.

KEZDJÜNK HOZZÁ – A PRINT PARANCS

A PRINT parancs azt közli a számítógéppel, hogy az információt jelenítse meg a képernyőn. Számokat és szöveget (betűket) egyaránt ki lehet írni, azonban minden esetben speciális szabályok szerint kell eljárni, amelyeket a következőkben ismertetünk.

Számok kiírása

A PRINT parancs után írjuk be a kívánt számokat. Írja be a következőket a gépbe:

PRINT 5

Nyomja meg a RETURN billentyűt, mire az 5-ös szám megjelenik a képernyőn.
Most írja be a következőket, és nyomja meg a RETURN-t:

PRINT 5,6

Ennél a PRINT parancsnál a vessző azt jelzi a számítógépnek, hogy egy számnál többet kívánunk kiírni.
Ha a számítógép a PRINT utasításban vesszőt talál a számfüzérben, a vessző után következő számokat 10 üres karakterrel jobbra fogja írni az előző számtól. Ha nincs szüksége az üres karakterhelyekre, használjon pontosvesszőt (;) vessző helyett a PRINT parancsban. Ekkor a gép mindössze három üres karakterrel fogja jobbra írni a számokat. Írja be ezeket a példákat és figyelje meg, mi történik:

PRINT 5;6 **RETURN**
PRINT 100;200;300;400;500 **RETURN**

A kérdőjel mint a PRINT parancs rövid formája

A PRINT parancs lerövidítésére használhatja a kérdőjelet (?) is. A könyv példái közül soknál használjuk a kérdőjelet a PRINT szó helyett. A legtöbb BASIC parancs egyébként rövidíthető, a rövidítéseket a könyv K) függelékében találja.

Szöveg kiírása

Most, hogy már tud számokat kiírni, itt az ideje, hogy megtanulja a szöveg kiírását is, ami egyébként igen egyszerű. A megjeleníteni kívánt karaktereket vagy szavakat idézőjelben kell beírni. *Füzérnek nevezzük BASIC-ben az idézőjelbe tett tetszőleges karaktersort.* Az idézőjelet úgy kell használni, hogy lenyomja a SHIFT billentyűt és a 2-es szám billentyűjét a felső sorban (nem a numerikus billentyűzeten). Próbálja ki a következő példákat:

? "COMMODORE 128" **RETURN**
? "4*5" **RETURN**

Figyelje meg, hogy ha megnyomja a RETURN-t, a számítógép kiírja az idézőjelbe tett számokat. A második példában a gép nem számolta ki a 4*5-öt, mert füzérként kezelte, nem pedig matematikai számításként. Ha ki akarja számíttatni a 4*5 eredményét, a következő parancsot használja:

? 4*5 **RETURN**

Bármilyen füzért kiírathat, ha a PRINT parancsot használja és a megjeleníteni kívánt karaktereket idézőjelbe teszi. A szöveget és a számításokat kombinálni is lehet egyetlen PRINT parancson belül:

? "4*5=" 4*5 **RETURN**

Figyelje meg, hogy a gép kiírja az idézőjelbe tett karaktereket, elvégzi a számítást és kiírja az eredményt. Annak, hogy a szöveg vagy a számítás áll-e elöl, nincs jelentősége. Mindkettőt többször is lehet használni egyetlen PRINT parancsban. Írja be a következő utasítást:

? 4*(2+3) " annyi mint " 4*5 **RETURN**

Figyelje meg, hogy az idézőjelen belül még az üres karakterhelyek is megjelennek a képernyőn. Pl.:

? "   IDE" **RETURN**

Színek megjelenítése a képernyőn

A Commodore 128-as számítógép 16 különböző szín megjelenítésére képes. A színek váltása igen könnyű feladat. Mindössze lenyomva kell tartani a CTRL billentyűt, és közben a főbillentyűzet felső sorában le kell nyomni egy megszámozott billentyűt 0 és 8 között. Figyelje meg, hogy a kurzor a megnyomott billentyűnek megfelelően színt vált. Az összes következő karakter a kiválasztott színben fog megjelenni. Ha lenyomva tartja a C= billentyűt és 0-tól 8-ig megnyom egy számozott billentyűt, 8 további szín jelenik meg a képernyőn.

A *3.1/a és b táblázat* felsorolja a C128-as módban használható színeket, 40 és 80 oszlopos képernyő esetén. Leolvasható róla az adott szín kiválasztásához szükséges billentyűk sorrendje is (CONTROL billentyű plusz számbillentyű vagy C= billentyű és számbillentyű).

3.1/a táblázat

A színek számai 40 oszlopos képernyőformátum esetén

Színkód	Szín	Színkód	Szín
1	Fekete	9	Narancs
2	Fehér	10	Barna
3	Vörös	11	Világosvörös
4	Cián	12	Sötétszürke
5	Bíbor	13	Középszürke
6	Zöld	14	Világoszöld
7	Kék	15	Világoskék
8	Sárga	16	Világosszürke

3.1/b táblázat

A színek számai 80 oszlopos képernyőformátum esetén

Színkód	Szín	Színkód	Szín
1	Fekete	9	Sötétbíbor
2	Fehér	10	Sötétsárga
3	Sötétvörös	11	Világosvörös
4	Világoscián	12	Sötétcián
5	Világosvörös	13	Középszürke
6	Sötétzöld	14	Világoszöld
7	Sötétkék	15	Világoskék
8	Világossárga	16	Világosszürke

A KURZORBILLENTYŰK HASZNÁLATA IDÉZŐJELBEN PRINT PARANCSNÁL

Ha idézőjelen belül üti le a kurzorbillentyűket, grafikus karakterek jelennek meg a képernyőn. Ezeket a karaktereket a RETURN billentyű megnyomása után a gép *nem* fogja kiírni. Próbálja ki a következőt: írjon be egy kérdőjelet (?), nyissa meg az idézőjelet (SHIFT plusz a 2-es billentyű), majd nyomja meg az alsó kurzorbillentyűk egyikét tízszer, utána írja be az ITT LENT szavakat és zárja az idézőjelet. A sor így fog kinézni:

? "◙◙◙◙◙◙◙◙◙◙ ITT LENT"

Most nyomja meg a RETURN billentyűt. A gép kihagy 10 üres sort és a 11. sorba kiírja: ITT LENT. Tehát, ha a kurzorbillentyűket idézőjelen belül használjuk, közölhetjük a számítógéppel, hogy hova írjon a képernyőn.

A PROGRAMOZÁS KEZDETE

Az idáig tárgyalt parancsok többségét direkt módban alkalmaztuk. Ez azt jelenti, hogy a RETURN billentyű megnyomása után a gép végrehajtotta a parancsot. A legtöbb BASIC parancsot és funkciót azonban program keretében is lehet használni.

Mi a program?

A program számozott BASIC utasítások sora, amely közli a számítógéppel, hogy használója mit akar tőle. Ezeket a számozott instrukciókat *utasítások*nak vagy *sorok*nak nevezzük.

Sorszámok

A program sorai meg vannak számozva, hogy a számítógép tudja, milyen sorrendben akarjuk

végrehajtani, azaz futtatni őket. A gép a számok sorrendjében hajtja végre a sorokat, hacsak a program nem rendelkezik másként. Sorszám bármilyen egész szám lehet 1-től 63 999-ig. *Soha ne használjunk vesszőt sorszámban.*
Sok, direkt módban már megtanult parancsot könnyűszerrel át lehet alakítani programutasítássá. Pl. írja be a következőket:

10 ? "COMMODORE 128" **RETURN**

Figyelje meg, hogy a számítógép nem írta ki a COMMODORE 128-at a RETURN billentyű megnyomása után, mint akkor, amikor a PRINT parancsot használtuk direkt módban. Ez azért van, mert a PRINT-et helyettesítő kérdőjel előtti szám azt közli a géppel, hogy BASIC programot akarunk rajta készíteni. Így a gép csak tárolja a számozott utasítást és várja a következő bevitelt.
Most írja be a RUN utasítást, és nyomja meg a RETURN billentyűt. A gép most kiírja a COMMODORE 128-at. Ez nem ugyanaz, mint a PRINT parancs használata direkt módban. *Az történt, hogy épp az imént írta meg és futtatta első BASIC programját,* bármilyen rövid volt is. A program még mindig benne van a számítógép tárában, tehát annyiszor futtathatja, ahányszor csak akarja.

A program megtekintése – a LIST parancs

Egysoros programja még mindig benne van a C128-as tárában. Most törölje a képernyőt a SHIFT és a CLR/HOME billentyűk együttes lenyomásával. A képernyő üres. Ilyenkor előfordulhat, hogy látni szeretné, hogy a program benne van-e még a gép tárában. A BASIC nyelvben van egy parancs, amely erre való, a LIST.
Írja be a LIST parancsot és nyomja meg a RETURN-t. A gép így válaszol:

10 PRINT "COMMODORE 128"
READY.

Ha bármikor látni akarja a program valamennyi sorát, írja be a LIST parancsot. Ez különösen akkor nagy segítség, ha változtatni kíván valamit, mert így ellenőrizni lehet, hogy az új sorok bekerültek-e a gép tárába. Válaszul a számítógép kiírja a sor, a sorok vagy a program megváltoztatott alakját. A LIST parancs használatának néhány szabálya:
- Ha csak az N. sort akarja látni, írja be a LIST N parancsot, és nyomja meg a RETURN-t. Az N helyére a látni kívánt sor számát írja be.
- Ha N-től végig kíváncsi a programra, írja be a LIST N- parancsot, majd nyomja meg a RETURN billentyűt.
- Ha a program elejétől az N. sorig kívánja látni a sorokat, írja be: LIST–N, és nyomja meg a RETURN-t.
- Ha N1-től N2-ig bezárólag akarja a sorokat látni, írja be a LIST N1–N2 parancsot és nyomja meg a RETURN-t.

Egyszerű ciklus – a GOTO utasítás

A program sorszámainak – azon kívül, hogy a gép számára a parancsokat megfelelő sorrendbe teszi – van egy másik funkciójuk is. Hivatkozási számként szolgálnak a gépnek, ha Ön egy bizonyos sorban szereplő parancsot ismételten alkalmazni kíván a program során. A GOTO parancscsal közölhetjük a számítógéppel, hogy menjen egy bizonyos sorba, és hajtsa végre az ott szereplő parancso(ka)t. Írja be:

20 GOTO 10

Ha a 20-as sor beírása után megnyomja a RETURN-t, a gép a tárban hozzáveszi a sort a programhoz.

Bizonyára észrevette, hogy az első sornak a 10-es, a másodiknak pedig a 20-as sorszámot adtuk. Ajánlatos a sorokat tízesével beszámozni (azaz 10, 20, 30, 40 stb.), mert előfordulhat, hogy később vissza akar menni és új sorokat betenni a már meglevők közé. Az ilyen pótlólag beírt sorokat lehet ötösével (15, 25 stb.) vagy egyesével (1, 2 stb.) számozni, bármilyen egész számmal, hogy a sorok helyes rendben maradjanak (l. a BASIC kislexikon RENUMBER és AUTO parancsait).
Írja be a RUN parancsot, nyomja meg a RETURN billentyűt, és megfigyelheti, hogy a COMMODORE 128 felirat elindul lefelé a képernyőn. Hogy a kiírást megállítsuk, nyomja meg a billentyűzet bal oldalán található RUN/STOP billentyűt.
Az Ön által beírt két sor egyszerű programot alkot, amely vég nélkül ismétlődik, mert a második sor állandóan visszautal az elsőre. A gép addig folytatja a kiírást, ameddig Ön meg nem állítja, vagy a gépet ki nem kapcsolja.
Most írja be, hogy LIST **RETURN**. A képernyőn a következő üzenetnek kell megjelennie:

10 PRINT "COMMODORE 128"
20 GOTO 10
READY

A program még mindig bent van a tárban. Ha akarja, újra lehet futtatni. Ez fontos különbség a PROGRAM mód és a DIREKT mód között. Ha a gép egy parancsot direkt módban egyszer végrehajtott, többé már nincs benne a tárban. Valószínűleg felfigyelt arra, hogy bár a PRINT utasítás-

ban kérdőjelet használt, a gép átváltoztatta és kiírta PRINT alakban. A gép listázásakor így tesz minden rövidített alakkal.

A számítógép tárának törlése – a NEW parancs

Ha mindent elölről akar kezdeni, és ki akarja törölni BASIC programját a gép tárából, írja be a NEW parancsot és nyomja meg a RETURN billentyűt. Ez a parancs törli a gép BASIC tárát, amelyben a programokat tárolja.

Színek használata a programban

Ha a programban színekkel akar dolgozni, a PRINT utasításon belül a színkiválasztásról is informálni kell a gépet. Pl. törölje a számítógép tárát a NEW paranccsal és a RETURN billentyű megnyomásával, majd írja be a következőket, vigyázva, hogy a betűk között üres karakterhelyet hagyjon:

10 PRINT " S P E C T R U M" **RETURN**

Most írja le ismét a 10-es sort, de ezúttal tartsa lenyomva a CTRL billentyűt és nyomja meg az 1-es számbillentyűt közvetlenül az első idézőjel beírása után. Engedje föl a CTRL billentyűt és írja be az S-t. Most tartsa ismét lenyomva a CTRL billentyűt és üsse le a 2-es számbillentyűt. Engedje el a CTRL-t és írja be a P-t. Tartsa lenyomva a CTRL billentyűt, és nyomja meg a 3-as billentyűt. Folytassa az előbbiek szerint, amíg le nem írta a SPECTRUM szó összes betűjét, minden betű között színt váltva. A SHIFT-tel és a 2-es billentyűvel zárja az idézőjelet, és nyomja meg a RETURN-t. Írja be a RUN parancsot és nyomja meg a RETURN billentyűt. A képernyőn a SPECTRUM szó minden betűje más színben jelenik meg. Most írja be a LIST-et és nyomja meg a RETURN billentyűt. A 10-es sorban a PRINT utasításnál grafikai karakterek jelennek meg. Ezek közlik a számítógéppel, hogy milyen színt kíván adni az egyes betűknek. Ha a Commodore 128-as kiírja a SPECTRUM szót más és más színű betűkkel, ezek a karakterek nem jelennek meg.
A színváltó karakterek a 10-es sor PRINT utasításában közlik a géppel, hogy váltson színt. A gép addig fogja az új színt használni, amíg nem találkozik egy másik színváltó karakterrel. Az idézőjelbe tett karaktereket a gép általában változatlanul kiírja, a vezérlőkaraktereket csak listázásnál.

A PROGRAM SZERKESZTÉSE

A következőkben programjai beírásához, javításához és kibővítéséhez adunk tanácsokat.

Egy sor törlése a programból

A LIST paranccsal írassa ki a képernyőre az előzőleg beírt programot. Most írja be a 10-es sorszámot, és nyomja meg a RETURN-t, ezáltal kitörölte a 10-es sort a programból. Listázza ki, és meggyőződhet róla. Ha a régi 10-es sor még mindig ott lenne a képernyőn, vigye föl a kurzort úgy, hogy tetszőleges helyen, de abban a sorban villogjon. Ha most megnyomja a RETURN billentyűt, a 10-es sor ismét visszakerül a számítógép tárába.

Egy sor másolása

Tartsa lenyomva a SHIFT billentyűt és nyomja meg a CLR/HOME billentyűt, amely a billentyűzet jobb felső részén található. Ezáltal a képernyő kiürül. Most listázza ki a programot. Vigye föl a kurzort, hogy a 10-es sor 0-ján villogjon. Írja be az 5-ös számot és nyomja meg a RETURN billentyűt. Most megduplázta (azaz lemásolta) a 10-es sort. Ez a másolt sor a 15-ös sorszámot viseli. Írja be a LIST parancsot és nyomja meg a RETURN-t, hogy lássa a másolt sort a programjában.

Egy sor kicserélése

Egy egész sort ki lehet cserélni úgy, hogy a régi sorszám után beírja az új sor szövegét, majd megnyomja a RETURN billentyűt. A sor korábbi változata törlődik a gép tárából, és mihelyt a RETURNT-t megnyomta, kicserélődik az új sorral.

Egy sor megváltoztatása

Előfordulhat, hogy valamit be akar szúrni egy sor közepébe. Egyszerűen vigye a kurzort arra a karakterre vagy üres karakterhelyre, amely közvetlenül az után a hely után jön, ahova az új anyagot be akarja szúrni. Tartsa lenyomva a SHIFT és az INST/DEL billentyűt egyszerre, amíg elegendő helye nem lesz az új karakterek beírásához.
Próbálja ki a következő példát. Írja be a NEW parancsot és nyomja meg a RETURN-t, hogy törölje a gép tárából az előző programot. Írja be:

10 ? "A 128-AS REMEK KIS MASINA"
RETURN

Mondjuk, hogy most be akarja írni a 128 elé a COMMODORE szót. Csak vigye a kurzort a 128-as szám 1. számjegyére. Tartsa lenyomva a SHIFT és az INST/DEL billentyűket, amíg a COMMO-

DORE szónak elég helye nem lesz (ne feledje az E betű utáni üres karakterhelyet), majd írja be a COMMODORE szót.
Ha valamit ki akar törölni egy sorból (beleértve a fölösleges üres karakterhelyeket is), vigye a kurzort az eltávolítani kívánt anyagrészt követő karakterre. Nyomja le csak az INST/DEL billentyűt. A kurzor elindul balra, és addig törlődnek a karakterek vagy üres helyek, amíg az INST/DEL-t lenyomva tartja.

MATEMATIKAI MŰVELETEK

A PRINT parancsot olyan műveletek elvégzésére is használhatja, mint az összeadás, kivonás, szorzás, osztás és hatványozás. A műveletet a PRINT parancs után kell beírni.

Összeadás és kivonás

Írja be a következő példákat:

PRINT 6+4 **RETURN**
PRINT 50−20 **RETURN**
PRINT 10+15−5 **RETURN**
PRINT 75−100 **RETURN**
PRINT 30+40,55−25 **RETURN**
PRINT 30+40;55−25 **RETURN**

Bizonyára megfigyelte, hogy a negyedik számítás (75−100) negatív számot eredményezett, valamint azt is, hogy egyetlen PRINT paranccsal több művelet elvégzésére is utasítani lehet a számítógépet. Attól függően, hogy a kiírt eredményeket egymástól 10, avagy három üres karakternyi távolságra kívánja-e, használjon vesszőt vagy pontosvesszőt.

Szorzás és osztás

Keresse meg a csillaggal (*) jelzett billentyűt a billentyűzet jobb oldalán. Ez a Commodore 128-as által használt szorzójel. A törtjel (/), amely a jobb oldali SHIFT billentyű mellett található, az osztás jele. Próbálja ki a következő példákat:

PRINT 5*3 **RETURN**
PRINT 100/2 **RETURN**

Hatványozás

A hatványozás a hatványalap hatványkitevőre emelése. A szorzóbillentyű mellett található, felfelé mutató nyíllal jelzett billentyűt (↑) használjuk hatványozásra. Ha egy számot hatványozni akar, használja a PRINT parancsot, majd írja be az adott számot, a felfelé mutató nyilat, végül a kitevőt, ebben a sorrendben. Ha pl. ki akarja számítani 3-nak a négyzetét, írja be:

PRINT 3↑2 **RETURN**

A műveletek sorrendje

Az előbbiekben már láttuk, hogy ugyanabban a PRINT parancsban hogyan lehet kombinálni az összeadást és a kivonást. Ha a szorzást vagy az osztást akarja összeadással és kivonással kombinálni, lehet, hogy nem a várt eredményt kapja. Pl. írja be a következőt:

PRINT 4+6/2 **RETURN**

Ha azt várta, hogy 10-et fog osztani 2-vel, bizonyára meglepődve tapasztalta, hogy eredményül 7-et kapott. Ennek az az oka, hogy a számítógép a szorzási és osztási műveleteket előbb végzi el, mint az összeadást vagy a kivonást. Ezt úgy mondják, hogy a szorzás és az osztás előnyben részesül (prioritást élvez) az összeadással és a kivonással szemben. A műveletek beírásának sorrendje itt nem számít. Azt a sorrendet, amelyben a gép a matematikai műveleteket elvégzi, műveleti sorrendnek nevezzük.
A hatványozás megelőzi mind a négy többi műveletet. Pl. ha azt írja be:

PRINT 16/4↑2 **RETURN**

a Commodore 128-as 1-et ad válaszul, mert először négyzetre emeli a 4-et, mielőtt a 16-ot osztaná.

A zárójel használata a műveletek sorrendjének meghatározására

Azt is közölheti a Commodore 128-assal, hogy melyik műveletet kívánja elsőként elvégeztetni, ha azt a műveletet zárójelbe teszi a PRINT parancsban. Pl. az iménti első példában, ha azt akarjuk, hogy a gép az osztás előtt összeadjon, ezt írjuk be:

PRINT (4+6)/2 **RETURN**

Így megkapjuk a kívánt választ, az 5-öt.
Ha a második példában azt kívánja, hogy a gép előbb az osztást végezze el és csak azután a hatványozást, ezt írja be:

PRINT (16/4)↑2 **RETURN**

Most megkapja a kívánt eredményt, a 16-ot.
Ha nem használunk zárójelet, a számítógép az előbbi szabályok szerint végzi el a műveleteket. Ha egy számításban minden műveletnek egyenlő esélye van, a gép balról jobbra végzi el a számítást. Pl. írja be:

```
PRINT 4*5/10*6 RETURN
```

Miután itt a műveleteket a gép balról jobbra végzi, az eredmény 12 (4*5=20...20/10=2...2*6=12). Ha a 4*5-öt akarja osztani 10*6-tal, ezt kell beírnia:

```
PRINT (4*5)/(10*6) RETURN
```

Az eredmény most .333333333.

ÁLLANDÓK, VÁLTOZÓK ÉS FÜZÉREK

Állandók

Az állandók (konstansok) numerikus értékek, azaz értékük nem változik egy egyenletben vagy programban. A 3-as szám pl. állandó, mint minden szám. A következő utasítás azt mutatja be, hogyan használja a számítógép az állandókat:

```
10 PRINT 3
```

Függetlenül attól, hányszor végezteti el a géppel ezt a sort, a válasz mindig 3 lesz.

Változók

A változók olyan értékek, amelyek az egyenlet vagy a programutasítás során megváltoznak. Van a számítógép BASIC tárának egy olyan része, amely a programban használt karakterek (számok, betűk és jelek) részére van fenntartva. Úgy képzelje el ezt a tárat, mint egy sor raktári rekeszt, amelyben a számítógép a programról tárol tudnivalókat; ezt a változók tárának nevezzük. Írja be ezt a programot:

```
10 X=5
20 ?X
```

Most futtassa a programot és figyelje meg, hogyan írja ki a gép ezt az 5-öst a képernyőre. A 10-es sorban azt mondta a gépnek, hogy az X betű az 5 számot fogja képviselni a program folyamán. Az X-et változónak nevezzük, mert értéke az egyenlőségjel jobb oldalán levő értéktől függően változik. Ezt hozzárendelő utasításnak nevezzük, mivel a számítógép tárában most van egy rekesz, amely az X címkét viseli, és ehhez az 5-ös számot hozzárendeltük. Az egyenlőségjel (=) azt közli a számítógéppel, hogy bármi következik tőle jobbra, a tőle balra álló X-szel jelölt rekeszhez kell hozzárendelni (tárkijelölés).
Az egyenlőségjel bal oldalán álló változó neve állhat egy vagy két betűből, vagy egy betűből és egy számból, de ilyenkor mindig a *betűnek kell elöl állnia*. A nevek lehetnek hosszabbak is, de a gép csak az első két betűt nézi. Ez azt jelenti, hogy a DANI és a DA ugyanarra a rekeszre vonatkozik. Fontos, hogy a BASIC parancsoknál használatos szavak (LOAD, RUN, LIST stb.) és a függvények (INT, ABS, SQR stb.) nem használhatók nevekként a programban. Lapozza fel az V. fejezetben a BASIC kislexikont, ha bármi kételye van afelől, hogy egy változó neve BASIC kulcsszó-e. Ne feledje, hogy a hozzárendelő utasításban az egyenlőségjel nem azonos a matematikai egyenlőséggel, hanem egy változó kijelölését és egy érték hozzárendelését jelenti.
Az előbbi mintaprogramban az X változó értéke mindig 5 marad. Az egyenlőségjel jobb oldalára tehetünk számításokat is, és az eredményt rendeljük egy változóhoz. Azonosításukra a szöveget lehet állandókkal keverni a PRINT utasításban. Írja be a NEW parancsot és nyomja meg a RETURN-t, hogy törölje a gép tárát (memóriáját), majd próbálja ki ezt a programot:

```
10 A=3*100
20 B=3*200
30 ?"A ANNYI MINT" A
40 ?"B ANNYI MINT" B
```

Most tehát két változó (A és B) található a számítógép tárában, az egyik a 300, a másik a 600 értéket tartalmazza. Ha később, a program folyamán meg akarja változtatni egy változó értékét, mindössze annyit kell tennie, hogy egy másik hozzárendelő utasítást ír be a programba. Az előző programhoz írja hozzá a következőket és futtassa újra:

```
50 A=900*30/10
60 B=95+32+128
70 GOTO 30
```

A programot most már csak a STOP gombbal állíthatja meg.
Most listázza ki a programot, és kövesse nyomon a gép lépéseit. Először is, a 10-es sorban az A-hoz rendeli az egyenlőségjeltől jobbra álló értéket. Ugyanezt teszi a 20-as sorban is a B-vel. Ezután a 30-as és 40-es sorban levő üzeneteket kiírja, amelyek megadják az A és a B értékét. Végül, az 50-es és a 60-as sorban új értékeket rendel az A-hoz és a B-hez. A régi értékek helyett most más áll, és amíg a gép a 10-es és a 20-as

sorokat nem hajtja végre, ez így is marad. Amikor a gépet elküldjük a 30-as sorba, hogy írja ki ismét az A és a B értékét, az 50-es és a 60-as sorban kiszámított új értékeket fogja kiírni. Az 50-es és a 60-as sor újra hozzárendeli ezeket az értékeket az A-hoz és a B-hez, a 70-es sor pedig visszaküldi a gépet a 30-asba. Ezt végtelen ciklusnak nevezzük, mert a 30–70-es sorokat a gép újra és újra végrehajtja, egészen addig, amíg meg nem nyomjuk a RUN/STOP billentyűt, hogy leállítsuk a programot. A cikluskészítés egyéb módszereit e fejezet további része, valamint a következő két fejezet tartalmazza.

Füzérek

Füzérnek egy idézőjelbe tett karaktert vagy karaktercsoportot nevezünk. Ezeket a karaktereket a számítógép tára igen hasonlóan tárolja, mint a numerikus változókat. Változók nevével ugyanúgy lehet füzéreket jelölni, mint számokat. Amikor dollárjelet ($) írunk a füzérváltozó neve után, ez a jel azt közli a számítógéppel, hogy a név nem numerikus, hanem füzérváltozóra vonatkozik.

Írja be a NEW parancsot és nyomja meg a RETURN-t, hogy ezáltal törölje a gép tárát, majd írja be a következő programot:

```
10 A$="COMMODORE""
20 X=128
30 B$=" SZÁMÍTÓGÉP"
40 Y=1
50 ? "A""A$;X;B$;Y" OSZTÁLYÚ"
```

Látja, hogyan lehet numerikus és füzérváltozókat ugyanabba az utasításba kiírni? Kísérletezzen a változókkal, írjon saját rövid programokat!
A változók értékeit direkt módban is ki lehet írni, miután a program lefutott. Írja be: ?A$;B$;X;Y miután az előbbi programot lefuttatta, és győződjön meg arról, hogy a három változó értéke még mindig benne van a számítógép tárában.
Ha törölni akarja a BASIC tár ezen területét, de a programját érintetlenül kívánja hagyni, használja a CLR parancsot. Írja be, hogy CLR **(RETURN)**, és minden állandó, változó és füzér törlődik. Azonban, ha beírja a LIST parancsot, látni fogja, hogy a program még mindig a tárban van. A korábban említett NEW parancs törli mind a programot, mind a változókat.

MINTAPROGRAM

Most bemutatunk egy mintaprogramot, amely az imént tárgyalt eljárások és parancsok nagy részét tartalmazza. Ez a program három szám (X, Y és Z) átlagát számítja ki és kiírja a képernyőre az értékeket és átlagukat. Ön átszerkesztheti a programot, azaz megváltoztathatja a 10–30-as sorokban levő számításokat, hogy ezáltal a változóknak új értéket adjon. A 40-es sor összeadja a változókat, és elosztja 3-mal, hogy megkapja az átlagot. Figyeljen a zárójel használatára, mert ezzel közli a géppel, hogy osztás előtt adja össze a számokat.

Tipp: Ha egyazon utasításban több, mint egy pár zárójelet használ, ajánlatos összeszámolni a kezdő és bezáró (bal és jobb oldali) zárójeleket, hogy biztosan egyenlő számban szerepeljenek.

```
10 X=46
20 Y=72
30 Z=114
40 A=(X+Y+Z)/3
50
60 ? X;Y;"ÉS "Z;"ÁTLAGA "A;
70 GOTO 90
90 END
```

PROGRAMOK TÁROLÁSA ÉS ISMÉTELT FELHASZNÁLÁSA

Miután megírta saját programját, bizonyára szeretné maradandóan tárolni, hogy egy későbbi időpontban behívhassa és újra használhassa. Ehhez vagy egy Commodore lemezmeghajtó egységre, vagy egy Commodore 1530-as kazettás egységre (Datasette) van szüksége.
Meg kell tanulnia néhány parancsot, amelyekkel kapcsolatot tud teremteni a számítógép és a lemezmeghajtó egység vagy a Datasette között. Ezek a parancsok egy parancsszóból és az azt követő néhány paraméterből állnak. A paraméterek betűk, szavak vagy jelek egy bizonyos parancsban, és különleges információkat közölnek a számítógéppel (pl. a file neve vagy egy numerikus változó, amely az eszköz (periféria) számát jelöli). Egy parancsnak több paramétere is lehet. Pl. a lemezformáló parancs tartalmazza a lemez nevét, egy azonosító számot, vagy kódot és még néhány paramétert. Paramétereket majdnem minden BASIC programban használunk, ezek között vannak változók és állandók. A lemezmeghajtó egység és a C128-as lemezinformációs paraméterei a következők:
Lemezkezelő paraméterek:
– Lemeznév: a használó által megadott, tetszőleges, 16 karakterből álló, azonosításra szolgáló név.
– File-név: a használó által megadott, tetszőleges, 16 karakterből álló, azonosításra szolgáló név.

- Azonosítási szám: tetszőleges, két számjegyből álló szám.
- Meghajtó- (drive-) szám: 0-t kell használni szóló, és 1-et kettős lemezmeghajtó egység esetében.
- Eszköz- (periféria-) szám: az eszközök (perifériák) számára előre meghatározott szám, pl. a Commodore lemezmeghajtó egység száma 8.

A lemez formálása – a HEADER parancs

Ha új (üres) lemezen akarunk programokat tárolni, először elő kell készíteni a lemezt az új adatok fogadására. Ezt nevezzük a lemez formálásának. *Figyelem:* Mindig kapcsolja be előbb a lemezmeghajtó egységet, mielőtt lemezt akar behelyezni.
A formálás a lemezt sávoknak és szektoroknak nevezett részekre osztja. Létrejön egy lemezkatalógus vagy *tartalomjegyzék* is (directory). Ha egy programot tárolunk a lemezen, a program neve bekerül a tartalomjegyzékbe.

A Commodore 128-as kétféle formáló parancsot használ. Az egyiket csak C128-as módban lehet alkalmazni, a másikat C64-es és C128-as módban egyaránt. A következőkben ismertetjük a C128-as módban alkalmazandó lemezformáló parancsokat. A C64-es üzemmódról szóló III. fejezet bővebben tartalmazza a C64-es módban való programozás és lemezkezelés tudnivalóit.
A lemezformáló parancsot HEADER parancsnak nevezik. Van rövid és hosszú alakja. Üres (új) lemez formálására a következő hosszú parancsot kell alkalmazni:
HEADER "lemeznév", i. d. [,D lemezmeghajtó száma][,[ON]U eszközszám]

A HEADER szó után írja be a választott lemeznevet idézőjelben. 16 karakterig bármilyen nevet választhat. Ajánlatos olyan neveket választani, amelyek megkönnyítik a lemezen tárolt programok azonosítását.
A lemeznév után írjon be egy vesszőt, és az I betűt, majd egy két karakterből álló azonosítási kódot, utána ismét vesszőt. Az azonosító kódnak nem kell feltétlenül számokból állnia, választhat betűket is. Nem árt, ha folyamatos kódrendszert alkalmaz, pl. A1, A2, B1, B2. Ha szóló lemezmeghajtó egysége van, nyomja meg most a RETURN-t, mert a Commodore 128-as automatikusan feltételezi, hogy a meghajtó száma 0 és az eszközszám 8. Ezeket a paramétereket meg lehet változtatni, ha egynél több vagy kettős lemezmeghajtó egységet használ.
A parancs következő paramétere a lemezmeghajtó száma. Nyomja le a D billentyűt, és ha szóló lemezmeghajtó egysége van, nyomja le a 0 billentyűt, utána írjon be egy vesszőt. A kettős lemezmeghajtó egység kódja 0 és 1. Az eszközszám paramétere U betűvel kezdődik, tehát nyomja meg az U billentyűt, majd írja be a Commodore lemezmeghajtó egység eszközszámát, a 8-at.
A HEADER parancs hosszú alakjának példája:

HEADER "PROBA",IA1,D0,U8 **RETURN**

Ez a parancs formálja a lemezt, neve a tartalomjegyzékben PROBA, az azonosítási szám A1, a 0 kódszámú meghajtón, a 8-as egységen. Ha egyéb értéket nem adunk meg, a gép automatikusan a lemezmeghajtó egység 0, és az eszközszám 8 alapértékeit használja. Ez a HEADER parancs egy elfogadható hosszú alakja:

HEADER "ALFA",I23 **RETURN**

A HEADER parancsot használhatjuk akkor is, ha egy egyszer már használt lemezről minden adatot ki akarunk törölni, hogy ismét felhasználhassuk az új lemezzel azonos módon. Vigyázzunk, nehogy olyan lemezt töröljünk, amelyre egyszer még szükségünk lehet!

Ha a lemezt előzőleg már megformáltuk a HEADER parancs hosszú alakjával, akkor a parancs rövid alakját is használhatjuk.
A rövid (gyors) alak megsemmisíti a tartalomjegyzéket, a hosszú alakkal megegyező módon törli az összes adatot, de megtartja az előzőleg használt azonosító kódot. Egy példa a rövid HEADER parancsra:

HEADER "UJ PROGRAMOK" **RETURN**

Kimentés (SAVEing) lemezre

C128-as üzemmódben a következő parancsok bármelyikének felhasználásával lemezen tárolhatja a programjait:

DSAVE"PROGRAMNEV" **RETURN**
SAVE"PROGRAMNEV",8 **RETURN**

Mindkét parancs használható. Ne feledje, hogy a "DSAVE" karaktersort úgy is megjelenítheti a képernyőn, hogy megnyomja az F5 jelű funkcióbillentyűt, vagy úgy, hogy maga írja be a karaktersort. A programnév bármilyen tetszőleges név lehet, csak ne legyen 16 karakternél hosszabb. Ne felejtse el, hogy a programnevet idézőjelbe kell tenni. Ugyanazon a lemezen nem tárolhat két programot azonos néven. Ha mégis így tenne, a második programot a gép nem veszi be, és a lemez csak az elsőt fogadja el. A második példá-

ban a 8-as azt jelenti, hogy a 8-as eszközszámon mentette ki a programját. A DSAVE utasítás használatakor nincs szükség a 8-as számra, mert a számítógép automatikusan feltételezi, hogy a 8-as eszközszámot használja.

Kimentés kazettára

Ha a program tárolására Datesette-et használ, helyezzen be egy üres kazettát a magnóba, tekerje előre a kazettát, ha szükséges, és írja be a következőt:

SAVE "PROGRAMNEV" **RETURN**

Először a SAVE szót, majd a programnevet kell beírnia. A programnév tetszőleges, 16 karakternél nem hosszabb név lehet.
Megjegyzés: A képernyő a program kimentésének időtartamára kiürül, de visszatér a normál állapotba, ha a kimentési folyamatnak vége van.
A lemezzel ellentétben szalagra két programot is ki lehet menteni ugyanazzal a névvel. Azonban amikor újra betölti a számítógépbe, a szalagon sorrendben elsőként szereplő program fog betöltődni, tehát ne adjon azonos nevet két programnak.
Ha egyszer kimentett egy programot, azt vissza lehet tölteni a számítógép tárába, és bármikor újra futtatható.

Betöltés (LOADing) lemezről

Egy program betöltése azt jelenti, hogy a lemezről a program átmásolódik a számítógép tárába. Ha a tárban már volt egy BASIC program, mielőtt a LOAD parancsot megadta, úgy az automatikusan törlődik.
A következő C128-as parancsok bármelyikét használhatja, ha lemezről akar BASIC programot betölteni:

DLOAD"PROGRAMNEV" **RETURN**
LOAD"PROGRAMNEV",8 **RETURN**

Ne feledje, hogy C128-as módban úgy is kiírhatja a DLOAD funkciót, hogy megnyomja a SHIFT-et és az F1-et (tehát az F2 funkcióbillentyűt használja), vagy úgy, hogy beírja a betűket. A második példában a 8-as azt jelzi a számítógépnek, hogy a 8-as számú eszközről tölt. Ne feledje ugyanabban az alakban használni a programnevet, mint amikor kimentette a programot, ellenkező esetben a gép ezt a választ adja: FILE NOT FOUND (nem találom a file-t).
Ha a program be van töltve, írja be a RUN parancsot, hogy a gép futtassa a programot. A Commodore 128-asnak van egy speciális RUN parancsa, amelyet arra használunk, hogy egyetlen sorral töltsük és futtassuk a programot C128-as üzemmódban. Írja be, hogy RUN, majd a programnevet (file-nevet) idézőjelben:

RUN"ALFA" **RETURN**

Betöltés kazettáról

Kazettáról való töltés esetén írja be a következő parancsot:

LOAD"PROGRAMNEV" **RETURN**

Ha nem tudja a program nevét, ezt is beírhatja:

LOAD **RETURN**

és a szalagon tárolt következő program fog betöltődni. Mialatt a Datasette keresi a programot, a képernyő üres marad. Ha megtalálta a programot, megjelenik a

FOUND PROGRAMNEV

üzenet. Hogy most valóban be is töltődjön a program, nyomja meg a C- billentyűt.
A programok kezdetének azonosítására használja a Datasette-en található számlálót. Így, ha meg akar találni egy bizonyos programot, csévélje előre a szalagot a 000 állásból a program elejét jelző számra, és írja be:

LOAD **RETURN**

Ilyenkor nem kell megadni a programnevet. A program automatikusan betöltődik, ui. a szalagon következő lesz a keresett program.

Egyéb, lemezzel kapcsolatos parancsok

Egy program ellenőrzése

Ha ellenőrizni akarja, hogy egy program helyesen van-e kimentve, C128-as módban használja a következő parancsot:

DVERIFY"PROGRAMNEV" **RETURN**

Ha a számítógép programja azonos a lemezen találhatóval, a képernyőn az

OK

üzenet jelenik meg.
A VERIFY parancs szalagon tárolt programoknál is használható. Írja be:

VERIFY"PROGRAMNEV" **RETURN**

Ilyenkor nem kell beírni sem a vesszőt, sem az eszköz (periféria) számát.

A lemez tartalomjegyzékének (directory) *kiíratása*

C128-as üzemmódban a következőképpen tekintheti meg a lemezen tárolt programok tartalomjegyzékét:

DIRECTORY **RETURN**

Ilyenkor megjelenik a tartalomjegyzék listája. Egyszerűbb, ha az F3 funkcióbillentyűt használja, ekkor a számítógép kiírja a DIRECTORY szót és végrehajtja a programot.

Programjai kimentéséhez és betöltéséhez szükséges további felvilágosításokért vagy egyéb lemezzel kapcsolatos információkért forduljon a lemezmeghajtó egységgel vagy a DATASETTE-el együtt beszerzett kézikönyvhöz, továbbá lapozza fel az V. fejezetben található BASIC 7.0 kislexikonnak a LOAD és a SAVE parancshoz tartozó meghatározásait.

****

Most már van némi fogalma a BASIC nyelvről és a programozás alapelveiről. A következő alfejezet ezekre az elvekre épül, és megismerteti Önt a BASIC nyelvű programozás további parancsaival, függvényeivel és eljárásaival.

4. ALFEJEZET

Haladó BASIC programozás

Számítógépes döntések – Az IF...THEN utasítás
A kettőspont használata 34

Ciklusok – A FOR...NEXT parancs
Üres ciklusok – a program késleltetése 35
A STEP parancs 35

Adatbevitel
Az INPUT parancs 36
A GET parancs 36
Mintaprogram 37
A READ-DATA parancs 37
A RESTORE parancs 38
Tömbök használata 38

Szubrutinok programozása
A GOSUB-RETURN parancs 39
Az ON GOTO/GOSUB parancs 40

Tárhelyek használata
A PEEK és a POKE parancs 40

BASIC függvények
Mi a függvény? 41
Az INT (Integer = egész szám) függvény 41
Véletlenszámok előállítása – az RND függvény 41

Az ASC és a CHR$ függvény 42

Füzérek és számok átváltása (konvertálása) 42
A négyzetgyökfüggvény (SQR) 42
Az abszolútérték-függvény (ABS) 42

A STOP és a CONT (Continue = folytatni) parancs

A következőkben egy sor olyan hasznos BASIC parancs, függvény és technika használatáról lesz szó, amelyeket C128-as és C64-es üzemmódban egyaránt alkalmazhatunk.
Ezek a parancsok és függvények – különböző ciklusok és egymásba ágyazott ciklusok segítségével – ismételt műveletek programozását teszik lehetővé; értéktáblázatokat kezelnek, leágaznak vagy átugranak a program egy másik részére, visszatérnek, változó értékeket rendelnek egy mennyiséghez, és még számos más művelet elvégzésére alkalmasak. A könyv példákkal és mintaprogramokkal világítja meg a BASIC fogalmak működését és kölcsönhatását.

SZÁMÍTÓGÉPES DÖNTÉSEK – AZ IF...THEN UTASÍTÁS

Miután már tudjuk, hogyan lehet új értékeket adni a változóknak, a következő lépés az, hogy ezeken az új értékeken alapuló döntéseket hozatunk a számítógéppel. Erre való az IF...THEN (ha...akkor) utasítás. Azt közöljük a számítógéppel, hogy egy parancsot csak akkor hajtson végre, ha (IF) egy feltétel igaz (pl. X=5). A parancs, amelyet bizonyos feltétel megléte esetén el akarunk végeztetni a géppel, az utasításban a THEN (akkor) szó után következik.

Törölje a gép tárát a NEW beírásával és a RETURN billentyű lenyomásával, majd írja be a következő programot:

```
10 J=0
20 ? J,"COMMODORE 128"
30 J=J+1
40 IF J=5 THEN GOTO 60
50 GOTO 20
60 END
```

Nem kell már a STOP billentyűt használnia, ha egy ciklikus programot meg akar szakítani. Az IF...THEN utasítás azt közli a számítógéppel, hogy írja ki a COMMODORE 128-at és növelje (egyesével) a J-t, amíg J=5 igaz nem lesz. Ha az IF feltétel nem igaz, a számítógép a program következő sorára ugrik, függetlenül attól, hogy mi jön a THEN szó után.
Bizonyára észrevette az END parancsot a 60-as sorban. Ajánlatos a program utolsó sorába az END utasítást írni, ezáltal közölhetjük a géppel, hol kell az utasítások végrehajtását abbahagynia.
Az IF utasításban a következő összehasonlító jeleket használhatjuk:

Jel	*Jelentés*
=	egyenlő
>	nagyobb mint
<	kisebb mint
< >	nem egyenlő
> =	nagyobb/egyenlő
= <	kisebb/egyenlő

Ha számokkal dolgozunk, ezek az összehasonlítások a matematikailag várt módon működnek. Másképp kell azonban meghatározni, hogy egy füzér nagyobb, kisebb vagy egyenlő-e a másikkal. A füzérekről az V. fejezet BASIC 7.0 kislexikonában is lesz szó.
Az 5. alfejezetben található az IF...THEN elmélet kibővítése olyan BASIC 7.0 parancsokkal, mint a BEGIN BEND és az ELSE.

A kettőspont használata

A kettőspont igen hasznos eszköz a programozásban. Ugyanazon sorban levő két (vagy több) BASIC parancs elválasztására használható.
A kettőspont után álló utasításokat a számítógép balról jobbra, sorban hajtja végre. 160 karakterig (a sorszámot is beleértve) annyi utasítást tehetünk egy sorba, amennyit csak akarunk. Ez 40 oszlopos képernyőformátum esetén négy, 80 oszlop esetén két teljes képernyősornak felel meg.
A kettőspont segítségével jól kihasználhatjuk az IF...THEN utasítás THEN részét. Közölhetjük a számítógéppel, hogy ha a feltétel igaz, több parancsot is hajtson végre. Törölje a tárat, és írja be a következő programot:

```
10 N=1
20 IF N<5 THEN PRINT N;" KEVESEBB MINT
5";GOTO 10
30 ?N; "NAGYOBB VAGY EGYENLŐ 5"
40 END
```

Most cserélje ki a 10-es sort N=20-ra, és futtassa újból a programot. Ha N kevesebb, mint 5, egynél több utasítást is végre lehet hajtatni a számítógéppel. A THEN parancs után bármilyen utasítás(ok) állhat(nak). Ne feledje, hogy a GOTO 10-et addig nem végzi el a gép, amíg N<5 igaz. Azt a parancsot, amelyet a megjelölt feltétel teljesülésétől függetlenül is követni kell, külön sorba kell írni.

CIKLUSOK – A FOR...NEXT PARANCS

Az IF...THEN bemutatásánál használt mintaprogramban azt akartuk a számítógéptől, hogy írja ki a Commodore szót ötször. Ezt úgy értük el, hogy utasítottuk: növelje a J változót mindig 1-gyel, amíg a J értéke el nem éri az 5-öt. Ekkor befejeztük a programot. BASIC nyelven létezik ennek egy egyszerűbb megoldása is, a FOR...NEXT ciklus alkalmazásával:

```
10 FOR J=1 TO 5
20 ? "COMMODORE"
30 NEXT J
40 END
```

Ha beírja és futtatja ezt a programot és a kapott eredményt összehasonlítja az IF...THEN program eredményével, azt tapasztalja, hogy azonosak. Majdnem azonosak azok a lépések is, amelyeket a gép a két program végrehajtásakor megtesz. A FOR...NEXT ciklus igen széles körben alkalmazható. Használatával meg lehet határozni, hányszor végezzen el a gép egy műveletet. Kövessük nyomon az előbbi programban a számítógép lépéseit.
Először is, a gép a J változó értékét 1-nek veszi. A 10-es sor FOR utasításában az 5 azt közli a számítógéppel, hogy a FOR és a NEXT között minden utasítást hajtson végre mindaddig, amíg J 5-tel lesz egyenlő. Ebben az esetben ilyen csak egy van, a PRINT utasítás.

Miután a számítógép J-nek az 1 értéket adja, összehasonlítja az 1-et az 5-tel, hogy megvizsgálja, igaz-e a J=5 – az IF...THEN utasításhoz nagyban hasonló módon. Mivel a J=5 még nem igaz, a gép folytatja a programot és végrehajtja a PRINT utasítást. Ezután elmegy a következő (NEXT) J utasításhoz, amely azt közli vele, hogy menjen vissza a FOR-hoz. A FOR utasítás szerint 1-gyel növelnie kell a J-t, és össze kell hasonlítania J-t 5-tel, majd, ha J=5 még mindig nem igaz, folytatnia kell tovább. A ciklus öt lépése után J egyenlő lesz 5-tel. Ekkor a gép leugrik a közvetlenül a NEXT utasítás után álló sorba, és onnan folytatja. Ebben az esetben a folytatás az END parancs, tehát a program megáll.

Üres ciklusok – a program késleltetése

Mielőtt még továbbhaladna, meg kell tudnia valamit a ciklusokról és használatukról. A ciklust arra is lehet használni, hogy lelassítsa a gépet. (Mostanra már bizonyára tapasztalta, hogy milyen gyorsan hajtja végre a parancsokat.)

Ki tudja-e találni – mielőtt még lefuttatta volna – hogy mit fog ez a program csinálni?

```
10 A$="COMMODORE C128"
20 FOR J=1 TO 20
30 PRINT
40 FOR K=1 TO 1500
50 NEXT K
60 PRINT A$
70 NEXT J
80 END
```

Az történt, amit várt? A 40-es és 50-es sorban levő ciklus azt közli a számítógéppel, hogy számoljon 1500-ig, mielőtt végrehajtaná a program hátralevő részét. Ezt késleltető ciklusnak nevezzük, és gyakran igen nagy hasznunkra van. Mivel a program főciklusán belül foglal helyet, beágyazott ciklusnak nevezzük. Ezeket akkor használjuk, ha azt akarjuk, hogy a gép egy bizonyos sorrendben hajtson végre feladatokat, és az egész parancssorozatot néhányszor megismételje.
Az 5. alfejezetből megtudhatja, hogyan lehet az új BASIC 7.0 parancs, a SLEEP segítségével késleltetni a programot.

A STEP parancs

Azt is közölni lehet a számítógéppel, hogy bizonyos egységekkel növelje a ciklusváltozót (10-esével, 0.5-esével stb.). Ehhez a STEP parancsot kell használni, a FOR utasítással. Pl. ha azt akarja, hogy a gép 10-esével számoljon, írja be a következő programot:

```
10 FOR X=0 TO 100 STEP 10
20 ?X
30 NEXT
```

Valószínűleg megfigyelte, hogy a NEXT parancsban nincs szükség az X-re, ha egyszerre csak egy ciklussal dolgozunk – erről később majd még lesz szó. Továbbá nemcsak növelni, de csökkenteni is lehet a ciklusváltozót. Pl. az előbbi programban cserélje ki a 10-es sort erre:

```
10 FOR X=100 TO Ø STEP -10
```

és a gép 100-tól 0-ig tizesével fog visszaszámlálni. Ha a FOR utasításnál nem használja a STEP parancsot, a gép automatikusan 1-gyel növeli a ciklusváltozót.

A FOR...NEXT parancs részei a következők:

FOR	a ciklus kezdetét jelölő szó;
X	ciklusváltozó, bármilyen numerikus változó lehet;
1	kezdőérték, tetszőleges pozitív vagy negatív szám lehet;
TO	összeköti a kezdő- és a végértéket;
100	végérték, tetszőleges pozitív vagy negatív szám lehet;
STEP	azt jelzi, hogy nem 1 lesz a növekedés;
2	növekedés, bármilyen pozitív vagy negatív szám lehet.

ADATBEVITEL

Az INPUT parancs

Érték hozzárendelése változóhoz

A NEW parancs beírásával és a RETURN billentyű leütésével törölje a gép tárát, majd írja be és futtassa a következő programot:

```
10 K=10
20 FOR I=1 TO K
30 ? "COMMODORE"
40 NEXT
```

A 10-es sorban levő K értékét meg lehet változtatni aszerint, hogy hányszor akarjuk a ciklust elvégeztetni. Ezt még a beíráskor, a program futtatása előtt kell megtenni. Mi lenne, ha akkor is megmondhatná a gépnek, hogy hányszor hajtsa végre a ciklust, ha már fut a program? Más szóval, azt akarjuk, hogy a K változó értékét bármikor meg lehessen változtatni a program megváltoztatása nélkül. Ezt a képességet a géppel való kommunikációnak nevezzük. A számítógépet utasítani lehet, hogy kérdezze meg, hányszor kívánja a kezelője végrehajtatni a ciklust. Ehhez használjuk az INPUT parancsot. Cserélje ki a 10-es sort a következőre:

```
10 INPUT K
```

Ha most futtatja a programot, a számítógép ?-lel válaszol, hogy tudassa, várja a K értékét. Írja be a 15-öt és nyomja meg a RETURN-t. A gép 15-ször fogja végrehajtani a ciklust.

Prompt üzenetek (visszajelentkezés)

INPUT utasításnál a számítógép üzenetben is közölheti, milyen változóra vár. Írja át a 10-es sort így:

```
10 INPUT "KEREK K-NAK EGY ERTEKET";K
```

Ne felejtse el idézőjelbe tenni a kiírandó üzenetet, amelyet *prompt*nak nevezünk. Feltétlenül pontosvesszőt kell tennie a bezáró idézőjel és a K közé. A promptba bármilyen üzenetet beírhat, csak arra kell ügyelnie, hogy az INPUT utasítás beleférjen 2 képernyősorba, csakúgy, mint a többi BASIC utasítás.
Az INPUT utasítást füzérváltozókkal is lehet használni. Ugyanazok a szabályok érvényesek a füzérekre is, mint a numerikus változókra. Ne felejtsen el $ jelet használni a füzérváltozók azonosítására. A NEW paranccsal és a RETURN billentyű megnyomásával törölje a gép tárát, majd írja be a következő programot:

```
10 INPUT "HOGY HIVNAK";N$
20 ? "SZIA ",N$
```

Most futtassa a programot. Amikor a számítógép megkérdezi: HOGY HIVNAK, írja be a nevét, utána ne felejtse el megnyomni a RETURN billentyűt. Ha egy (numerikus vagy füzér) változó egyszer az INPUT használata útján belekerült a programba, a program során bármikor lehet rá hivatkozni a változó nevével. Írja be a ? N$ utasítást <**RETURN**>, és a számítógép megjegyzi a nevét.

A GET parancs

Más BASIC parancsok is vannak, amelyekkel egy program során beszélgetni tudunk a számítógéppel. Ilyen az INPUT-hoz hasonló GET parancs. A tár törlése után írja be a következő programot:

```
10 GET A$
20 IF A$="" THEN GOTO 10
30 ?A$
40 END
```

Ha ezt futtatja és megnyomja a RETURN-t, látszólag semmi sem történik. Ennek az az oka, hogy a számítógép egy billentyű megnyomására vár. A GET parancs arra utasítja a számítógépet, hogy nézze végig a billentyűzetet, és keresse meg, hogy melyik karakter, ill. billentyű van lenyomva. A gépnek kielégítő válasz az is, ha egyik sincsen (zéró karakter). Ezért van a 20-as sor, amelyből a gép azt tudja meg, hogyha zéró karaktert kap – amelyet a két, üres karakter (szóköz) nélkül beírt idézőjel jelöl ("") – menjen vissza a 10-es sorba és keressen egy másik billentyűt. Ez a ciklus addig ismétlődik, amíg egy lenyomott billentyűt nem talál. Ekkor a gép a billentyű karakterét hozzárendeli az A$-hoz.
A GET parancs igen fontos, mert ezzel lehet billentyűket beprogramozni. A következő példa eredményeképpen egy üzenet jelenik meg a képernyőn, ha lenyomjuk a Q billentyűt. Írja be a programot, és futtassa. Azután nyomja meg a Q-t és figyelje meg, mi történik.

```
10 ?"NYOMD MEG A Q-T, HA UZENETET
AKARSZ LATNI"
20 GET A$
30 IF A$=""THEN GOTO 20
40 IF A$ ="Q" THEN GOTO 60
50 GOTO 20
60 FOR I=1 TO 25
70 ?"MAR TUDOM HASZNALNI A GET
UTASITAST"
80 NEXT
90 END
```

Ha nem a Q-t nyomja le, hanem egy másik billentyűt, a gép nem írja ki az üzenetet, hanem visszamegy a 20-as sorba új karakterért.
Az 5. alfejezetben találja a DO/LOOP és a GETKEY utasításokat, amelyek hasonló feladatok elvégzésére valók, de új és hatásosabb BASIC 7.0 parancsok.

Mintaprogram

Most, hogy már tudja, hogyan kell a FOR...NEXT ciklust és az INPUT parancsot használni, törölje a gép tárát a NEW **(RETURN)** paranccsal, majd írja be a következő programot:

```
10 T=0
20 INPUT"HANY SZAM ATLAGAT KERI?";N
30 FOR J=1 TO N
40 INPUT"IRJON BE EGY SZAMOT ";X
50 T=T+X
60 NEXT
70 A=T/N
80 PRINT
90 ? "VAN OSSZESEN";N"SZAMA, MELYEK
OSSZEGE";T
100 ? "ATLAGUK = ";A
110 END
```

Ezzel a programmal közölheti a géppel, hogy hány számnak akarja kiszámítani az átlagát. A számokat bármikor kicserélheti anélkül, hogy a programot megváltoztatná.
Nézzük most meg sorról sorra, mit csinál a gép:

A 10-es sor T-nek 0 értéket ad (T a számok kurrens összege lesz).

A 20-as sor lehetőséget ad arra, hogy meghatározzuk, hány számnak akarjuk az átlagát venni, ezt a gép az N változóban tárolja.

A 30-as sor azt közli a számítógéppel, hogy N-szer hajtsa végre a ciklust.

A 40-es sorba kell beírnia a számokat, amelyeknek ki akarja számítani az átlagát.

Az 50-es sorban történik a számok összeadása.

A 60-as sor arra utasítja a számítógépet, hogy menjen vissza a 30-as sorba, emelje a ciklusváltozó értékét (J), és kezdje újból a ciklust.

A 70-es sorban a gép elosztja az összeget a beírt számok számával (N), miután N-szer végrehajtotta a ciklust.

A 80-as sor üres sort ír ki a képernyőre.

A 90-es sor kiírja az üzenetet, amely közli a számok darabszámát, valamint összegüket.

A 100-as sor kiírja a számok átlagát.

A 110-es sor közli a számítógéppel, hogy a programnak vége.

A READ-DATA parancs

Ezzel a paranccsal azt közölhetjük a számítógéppel, hogy milyen számokat, ill. karaktereket használjon a programban. A READ utasítás alkalmazásával a gép kiválaszt egy számot vagy karaktert a DATA utasításból. Pl. ha azt akarjuk, hogy a gép kiszámítsa öt szám átlagát, a következőképpen használjuk a READ és a DATA utasításokat:

```
10 T=0
20 FOR J=1 TO 5
30 READ X
40 T=T+X
50 NEXT
60 A=T/5
70 ? "AZ ATLAG = ";A
80 END
90 DATA 5, 12, 1, 34, 18
```

Ha futtatjuk a programot, a gép kiírja: AZ ATLAG=14. A program a T változót használja a folyamatos összeadás összegének jelölésére, és ugyanúgy számolja ki az átlagot, mint az INPUT átlagszámító program. A READ-DATA átlagszámító program az átlagolni kívánt számokat azonban a DATA sorban találja. Figyeljen a 30-as sorra: READ X. Itt a READ parancs azt közli a számítógéppel, hogy valahol a programban kell egy DATA utasításnak lennie. A gép megtalálja a DATA sort és az első számot veszi az X változó aktuális értékének. A cikluson belül a következő alkalommal a DATA utasításban szereplő második szám lesz az X értéke, és így tovább.
A DATA utasításba bármilyen számot be lehet tenni, számítást azonban nem. A DATA sor bárhol elhelyezhető a programban – még az END utasítás után is. Ez azért lehetséges, mert a gép a DATA utasítást soha nem hajtja végre, csak hivatkozik rá. Ne feledjen el vesszőt tenni a számok közé, de a DATA szó és az első szám közé soha ne kerüljön vessző.
Ha egy programban több mint egy DATA utasítás van, a gép arra hivatkozik, amelyik a legközelebb van a READ utasítás után. A gép egy mutatót használ, hogy vissza tudjon emlékezni, melyik adatot olvasta utoljára. Miután az elsőt beolvasta, a mutató a másodikra ugrik. Amikor a számítógép újra odaér a READ utasításhoz, a mutató

által jelzett értéket a READ utasításban megadott változó nevéhez utalja. Annyi READ és DATA utasítást használhat egy programban, amennyit csak akar, de arra ügyeljen, hogy legyen elég adat a DATA utasításokban. Az előbbi program DATA sorából töröljön egy számot, és futtassa újra a programot. A számítógép a ? OUT OF DATA ERROR IN 30 (kifogytam az adatokból, hiba a 30-as sorban) üzenettel válaszol. Az történt ui., hogy amikor a gép ötödszörre hajtotta végre a ciklust, már nem volt adat, amit be tudott volna olvasni. Ezt közli az előző hibaüzenettel. Ha túl sok adatot tesz a DATA utasításba, az nem jelent problémát, mert a számítógép soha nem jön rá, hogy felesleges adatok vannak a programban.

A RESTORE parancs

Ezt a parancsot arra használhatjuk, hogy ha szükséges, az adatmutatót visszaállítsuk vele az első adatra. Az előbbi program 80-as sorának END utasítása helyett írja be a következőt:

```
80 RESTORE
```

majd tegye hozzá:

```
85 GOTO 10
```

Ha futtatja a programot, azt fogja tapasztalni, hogy az folyamatosan fut és ugyanazt a DATA utasítást használja. *Ne feledje:* Ha a képernyőn megjelenik az OUT OF DATA ERROR hibaüzenet, annak az az oka, hogy a DATA utasításba elfelejtette visszatenni az előzőleg eltávolított számot, tehát már minden adatot felhasznált, mielőtt végrehajtotta volna a megadott számú READ utasítást.
DATA utasításokkal füzérváltozóknak is lehet értéket adni. Itt is ugyanazok a szabályok érvényesek, mint a numerikus adatok esetében. Törölje a tárat, és írja be a következő programot:

```
10 FOR J=1 TO 3
20 READ A$
30 ? A$
40 NEXT
50 END
60 DATA COMMODORE, 128, COMPUTER
```

Ha a READ utasítás füzérváltozóra vonatkozik, a DATA sorba betűket és számokat is lehet írni. Azonban, mivel a gép füzért olvas (READ), a számokat karakterfüzérként kezeli, nem pedig értékként, amellyel műveleteket végezhet. Az ilyen, füzérként tárolt számokat ki lehet íratni, de műveleteket nem lehet velük végezni. Ha a READ utasítás numerikus változóra vonatkozik, a DATA utasításba nem lehet betűket beírni.

Tömbök használata

Láttuk, hogyan használható a READ-DATA parancs, hogy egy változóhoz több értéket rendeljünk. De mit tegyünk akkor, ha azt akarjuk, hogy a gép a DATA utasítás minden adatára emlékezzen, ahelyett, hogy a változó értékét új adatokkal helyettesíti? Mit tegyünk olyankor, ha a harmadik számot vagy a második karakterfüzért akarjuk behívni?
Valahányszor a számítógép új értéket rendel egy változóhoz, törli a régi értéket a „változórekeszben" és helyette az újat tárolja. Lehet a számítógépet utasítani, hogy foglaljon le a tárban egy egész sor rekeszt, és tároljon minden egyes odarendelt változót. Ezt a rekeszsort nevezik tömbnek.

Indexelt változók

Ha a tömb az X változóhoz rendelt valamennyi értéket tartalmazza a READ-DATA példában, akkor X tömbnek nevezzük. Az X-nek adott első érték az X(1), a második az X(2) stb. Ezeket indexelt változóknak nevezzük, a zárójeles számokat pedig indexszámoknak. Index lehet egy változó vagy egy számítás is. Következik az átlagszámító program egy másik változata, ezúttal indexelt változókkal:

```
 5 DIM X(5)
10 T=0
20 FOR J=1 TO 5
30 READ X(J)
40 T=T+X(J)
50 NEXT
60 A=T/5
70 ? "AZ ATLAG = ";A
80 END
90 DATA 5,12,1,34,18
```

Látja, hogy nem történt sok változás. Az egyetlen új utasítás az 5-ös sor. Azt közli a számítógéppel, hogy tegyen félre a tárban 5 rekeszt az X tömb számára. A 30-as sor megváltozott úgy, hogy mindig, amikor a gép elvégzi a ciklust, a DATA utasításból egy értéket hozzárendel az X tömb J ciklusváltozónak megfelelő pozíciójához. A 40-es sor ugyanúgy összead, mint az előbb, de ehhez most indexelt változóra van szükség. Ha a program futtatása után szeretné visszahívni a harmadik számot, írja be a ?X(3) **RETURN** parancsot. A számítógép emlékszik az X tömb minden egyes számára. A füzérváltozók karaktereinek tárolására hasonlóképpen lehet füzértömböket is csinálni. Írja át a COMMODORE 128 COMPUTER READ-DATA programját úgy, hogy a számítógép emlékezzen az A$ tömb elemeire.

```
 5 DIM A$(3)
10 FOR J=1 TO 3
```

```
20 READ A$(J)
30 ?A$(J)
40 NEXT
50 END
60 DATA COMMODORE, C128, COMPUTER
```

Tipp: Ha a tömb 10 elemnél nem tartalmaz többet, a DIM utasítás felesleges, l. a „Tömbök dimenzionálása" c. részt.

Tömbök dimenzionálása

Az adatkezelés fejlettebb módja az, ha a tömböket egymásba ágyazott ciklusokkal együtt használjuk. Vegyünk pl. egy táblázatot, amelyben 10 sor van, mindegyikben öt számmal. Mondjuk, hogy minden sorban tudni szeretnénk az öt szám átlagát. Csinálhatnánk 10 tömböt, és kiszámíthatnánk külön az egyes sorok átlagát, ez azonban nem szükséges, mert a számokat kétdimenziós tömbbe is el lehet helyezni. Ennek a tömbnek ugyanazok a dimenziói, mint a feldolgozni kívánt számtáblázatnak – 10 sor és 5 oszlop. A tömbre (X tömb) vonatkozó DIM utasítás a következő:

```
20 DIM X(10,5)
```

Ez arra szólítja fel a számítógépet, hogy csináljon helyet a tárban egy X nevű kétdimenziós tömbnek. A gép ekkor 50 szám tárolásához biztosít elegendő helyet. Nem kell feltétlenül annyi számmal megtölteni a tömböt, amennyire méreteztük, de a gép akkor is fenn fog tartani helyet a tömb minden eleme számára.

Mintaprogram

Most már igen egyszerű hivatkozni a táblázat bármely számára, csak meg kell adni az oszlop- és sorszámát. A 10. sor 3. elemére a programban mint X(10,3) kell hivatkozni.

A következő program a táblázat számait kétdimenziós tömbbe olvassa be és soronként kiszámítja a számok átlagát.

	Oszlop				
Sor	1	2	3	4	5
1	1	3	5	7	9
2	2	4	6	8	10
3	5	10	15	20	25
4	10	20	30	40	50
5	20	40	60	80	100
6	30	60	90	120	150
7	40	80	120	160	200
8	50	100	150	200	250
9	100	200	300	400	500
10	500	1000	1500	2000	2500

```
10 REM **** P.1 ****
20 DIM X(10,5),A(10)
30 FOR R=1 TO 10
40 T=0
50 FOR C=1 TO 5
60 READ X(R,C)
70 T=T+X(R,C)
80 NEXT C
90 A(R)=T/5
100 NEXT R
110 FOR R=1 TO 10
120 PRINT"SORSZAM: ";R
130 FOR C=1 TO 5
140 PRINTX(R,C):NEXT C
150 PRINT"ATLAG = ";A(R)
160 FOR D=1 TO 1000:NEXT
170 NEXT R
180 DATA 1,3,5,7,9
190 DATA 2,4,6,8,10
200 DATA 5,10,15,20,25
210 DATA 10,20,30,40,50
220 DATA 20,40,60,80,100
230 DATA 30,60,90,120,150
240 DATA 40,80,120,160,200
250 DATA 50,100,150,200,250
260 DATA 100,200,300,400,500
270 DATA 500,1000,1500,2000,2500
280 END
```

SZUBRUTINOK PROGRAMOZÁSA

A GOSUB-RETURN parancs

Mostanáig, ha azt akartuk, hogy a számítógép ugorjon a program egy másik részére, a GOTO parancsot használtuk. Vajon mit kell tennünk akkor, ha azt akarjuk, hogy a gép ugorjon a program egy másik részére, hajtsa végre az ott található utasításokat, majd térjen vissza arra a pontra, ahonnan elindult, és folytassa a program végrehajtását?

A programnak azt a részét, ahová a gép elugrik, hogy végrehajtsa, *szubrutin*nak nevezzük. A tár törlése után írja be a következő programot:

```
10 A$="SZUBRUTIN":B$ ="PROGRAM"
20 FOR J=1 TO 5
30 INPUT "IRJON BE EGY SZAMOT";X
40 GOSUB 100
50 PRINT B$: PRINT
60 NEXT
70 END
100 PRINT A$: PRINT
110 Z=X↑2:PRINT Z
120 RETURN
```

Ez a program négyzetre emeli a beírt számot és kiírja az eredményt. A többi PRINT utasítás azt közli, hogy a gép mikor hajtja végre a szubrutint és mikor a főprogramot. A 40-es sor azt mondja a gépnek, hogy ugorjon a 100-as sorba, hajtsa végre azt, és az azt követő utasításokat egészen addig, amíg RETURN paranccsal nem találkozik. Ez ui. arra készteti a gépet, hogy menjen vissza a GOSUB parancsot követő sorba, és folytassa a munkát. A szubrutin akárhol helyet foglalhat a programban – még az END utasítás után is. Ne feledje, hogy a GOSUB és a RETURN utasításokat mindig együtt kell használni a programban (mint a FOR...NEXT és az IF...THEN utasításokat), egyébként a gép hibaüzenettel válaszol.

Az ON GOTO/GOSUB parancs

Másképpen is elérhetjük, hogy a program során a számítógép a program egy másik részére ugorjon (ágazzon le). Az ON utasítás használatával azt kérhetjük a számítógéptől, hogy számítás vagy billentyűs adatbevitel alapján döntse el, hogy a program melyik részére kell ugrania. Az ON utasítást együtt használjuk szükség szerint vagy a GOTO, vagy a GOSUB-RETURN paranccsal. A változó vagy a számítás mindig az ON után álljon, míg a GOTO vagy GOSUB után a sorszámok listájának kell következnie. Az ON parancs működtetésének érzékeltetésére írja be a következő programot:

```
 10 ? IRJON BE EGYTOL OTIG EGY SZAMOT"
 20 INPUT X
 30 ON X GOSUB 100,200,300,400,500
 40 END
100 ?"EGYEST IRT":RETURN
200 ?"KETTEST IRT":RETURN
300 ?"HARMAST IRT":RETURN
400 ?"NEGYEST IRT":RETURN
500 ?"OTOST IRT":RETURN
```

Ha X értéke 1, a gép a lista első számára ugrik, ha 2, a másodikra stb.

TÁRHELYEK HASZNÁLATA

A PEEK és a POKE parancs

A számítógép tárában minden területnek speciális funkciója van. Pl. nagy terület van fenntartva a programok és a velük kapcsolatos változók tárolására. A tárnak ezt a részét RAM-nak (Random Access Memory) nevezzük, és a NEW paranccsal törölhetjük. A többi terület nem ilyen nagy, azonban igen különleges funkciók ellátására alkalmas. Pl. van a tárhelyeknek egy területe, amely a számítógép zenei képességeit vezérli. Van két BASIC parancs – a PEEK és a POKE –, amelyekkel hozzá lehet férni a számítógép tárához és bele is lehet avatkozni. Használatuk igen hatékony programozási fogás, mert a gép tárhelyeinek tartalma határozza meg, hogy egy adott időpontban mit kell a gépnek csinálnia.

A PEEK használata PEEK utasításra a számítógép közli, hogy egy tárhelyen milyen értéket tárol (ez 0 és 255 között bármilyen érték lehet). Bármelyik tárhely – akár RAM akár ROM (Read Only Memory) – tartalmát ki lehet olvasni (PEEK = kiles) direkt és program módban egyaránt.

P=PEEK(2954) **RETURN**
?P **RETURN**

Ha az első sor után megnyomjuk a RETURN-t, a gép hozzárendeli a P változóhoz a 2594-es számú tárhelyen levő értéket. Ha beírjuk a ? P parancsot és lenyomjuk a RETURN-t, akkor ki is írja ezt az értéket. A 2594-es tárhely dönti el, hogy a szóköz- vagy a CRSR billentyűk ismétlődnek-e, ha lenyomva tartjuk őket. A 2594-es tárhelyen levő 128-as érték azt közli a számítógéppel, hogy a lent tartott billentyű működése ismétlődő legyen. Tartsa lent a szóközbillentyűt, és figyelje a kurzor mozgását a képernyőn.

A POKE használata Egy bizonyos RAM tárhelyen tárolt értéket a POKE paranccsal lehet megváltoztatni. Írja be a következőt:

POKE 2594,96

A számítógép a vessző után álló értéket (96) tárolja a vessző előtt álló tárhelyen (2594). A 96-os a 2594-es tárhelyen arra utasítja a gépet, hogy ne ismételje a lent tartott szóköz- vagy CRSR billentyű működését. Nyomja le a szóközbillentyűt és figyelje a kurzort. A kurzor egy helyet megy jobbra, de ez nem ismétlődik. Ha vissza akarja állítani a gépet a normál állapotába, írja be:

POKE 2594,128 **RETURN**

A gépnek nem az összes tárhelyén lehet megváltoztatni az értékeket, a ROM értékeket pl. csak leolvasni lehet, kicserélni nem.

Megjegyzés: Ezek a példák a bank 0 esetet feltételezik. Bővebben l. az V. fejezetben a BASIC 7.0 kislexikon BANK címszavát. A Commodore 128 Programmer's Reference Guide (C128-as programozói kézikönyv) a gép komplett tártérképét tartalmazza, és közli az összes tárhely tartalmát.

BASIC FÜGGVÉNYEK

Mi a függvény?

A függvény a BASIC nyelv előre meghatározott művelete, amely általában egyetlen értéket ad. Ezt úgy nevezzük, hogy a gép „megadja" (visszaadja – returns) az értéket. Pl. a SQR (square = négyzetgyök) függvény egy olyan matematikai függvény, amely gyökvonással „visszaadja" a négyzetre emelt szám eredeti értékét.

Kétféle függvény van:

numerikus: egyetlen számból álló eredményt ad, amely lehet valamely matematikai érték kiszámításától a tárhely numerikus értékének megadásáig bármilyen szám;
füzér: karakter eredményt ad.
A következőkben ismertetjük a leggyakrabban használt függvényeket. A BASIC 7.0 függvények teljes listáját megtalálja a BASIC 7.0 kislexikonban.

Az INT (INTEGER = egész szám) függvény

Ha egy számot a legközelebbi egész számra akarunk kerekíteni, az INT függvényt kell használnunk. Ez a tizedespont után semmit sem vesz figyelembe. Írja be a következő példákat:

? INT (4.25) **RETURN**
? INT (4.75) **RETURN**
? INT(SQR(50)) **RETURN**

Ha a legközelebbi egész számra akarunk kerekíteni, akkor a második példában az eredménynek 5-nek kellene lennie. Ténylegesen 5 tizeden felül mindent felfelé kell kerekíteni. Ehhez az INT függvény használata előtt 0.5-öt hozzá kell adni a számhoz, így a tizedespont utáni részükben 5 tizednél nagyobb értéket tartalmazó számok 1-gyel nőnek, mielőtt az INT függvény lekerekítené őket. Pl.:

? INT (4.75 + 0.5) **RETURN**

Az INT függvény elvégzése előtt a 4.75-höz a gép hozzáadott 0.5-öt, így végül 5.25-öt kerekített le 5-re. Ha egy számítási művelet végeredményét akarjuk kerekíteni, ezt kell tenni:

? INT ((100/6) + 0.5) **RETURN**

A belső zárójelben szereplő osztás helyett természetesen bármilyen szám állhat.
Ha a legközelebbi századra akarjuk a számokat kerekíteni, 0.5 helyett adjunk hozzá 0.005-öt, és szorozzuk meg 100-zal. Mondjuk, hogy a 2.876-ot századra akarjuk kerekíteni. Az előbbi módszer alapján így kezdjünk hozzá:

?(2.876 + 0.005)*100 **RETURN**

Most használjuk az INT függvényt, amely a tizedespont után semmit sem vesz figyelembe (a szorzás eredményeképpen a tizedespont két hellyel jobbra megy). Az eredmény:

? INT ((2.876 + 0.005)*100) **RETURN**

amelynek értéke 288. Most már ezt csak osztani kell 100-zal és megkapjuk a kívánt 2.88 értéket. Ezzel a módszerrel pl. a következő példához hasonló számítások eredményét is lehet századokra kerekíteni:

? INT((2.876 + 1.29 + 16.1 − 9.534) + 0.005)*
100/100 **RETURN**

Véletlenszámok előállítása – az RND függvény

Az RND függvénnyel a számítógép véletlenszámokat tud előállítani. Ezt szerencsejátékok szimulálására, valamint érdekes grafikai vagy zenei programok készítésére használhatjuk. Minden véletlenszám (RND = random) kilenc számjegyű, decimális, és a 0.000000001 és a 0.999999999 között foglal helyet. Írja be a következőt:

? RND (0) **RETURN**

Ha a véletlenszerűen előállított számot 6-tal megszorozzuk, a számok 0-nál nagyobbak és 6-nál kisebbek lesznek. Ha a 6-ot is fel kívánjuk venni az előállított számok közé, az eredményhez (RND(0)*6) hozzá kell adni 1-et. Így a tartomány 1 < X < 7 lesz. Ha a tizedeshelyek kiküszöbölésére az INT függvényt használjuk, a paranccsal elő tudunk állítani egész számokat 1 és 6 között. Ezt a folyamatot a kockadobás szimulálására használhatjuk. Próbálja ki ezt a programot:

```
10 R=INT(RND(1)*6+1)
20 ?R
30 GOTO 10
```

Minden előállított szám egy kockadobásnak felel meg. Egy kockapár dobásainak szimulálásához két ilyen parancsot kell alkalmazni. Minden számot külön állít elő a gép, és a két szám összege adja a „dobott" eredményt.
Az 5. alfejezetben leírt DO/LOOP utasítással más módon is lehet véletlenszámokat előállítani.

Az ASC és a CHR$ függvény

Minden karakterhez, amelyet a Commodore 128-as ki tud írni (beleértve a grafikus karaktereket is) egy szám van hozzárendelve, amelyet *karakterfüzér kód*nak (CHR$) nevezünk, és amiből a Commodore 128-as gép esetében 255 van. Ezzel kapcsolatban két jól használható függvényt mutatunk be, az első az ASC függvény.

Írja be a gépbe a következőt:

ASC(„Q") **RETURN**

A gép válaszul kiírja a 81-et. A 81 a Q billentyű karakterkódja. Ha bármilyen más karakterrel helyettesíti a Q-t, megtudhatja az illető karakter ASCII kódszámát.
A második a CHR$ függvény, amely az előbbi fordítottja. Mindkettő a gép tárában levő karakterkód táblázatra vonatkozik. A CHR$ értékekkel lehet beprogramozni a funkcióbillentyűket. A CHR$ használatáról bővebbet az 5. alfejezetben olvashat.
Az E) függelék tartalmazza az ASC és CHR$ kódok teljes listáját.

Füzérek és számok átváltása (konvertálása)

Előfordulhat, hogy füzérváltozókként tárolt numerikus karakterekkel kell számításokat végeznie, vagy a számokon füzér műveleteket végrehajtania. Van két BASIC függvény, amellyel a numerikus és füzérváltozókat oda-vissza lehet átváltani.

A VAL függvény A VAL függvény füzér helyett numerikus értéket ad meg. Törlés után írja be ezt a programot:

```
10 A$= "64"
20 A=VAL(A$)
30 ? A$;"ERTEKE";A
40 END
```

A STR$ függvény Ez az előbbi fordítottja, valamely numerikus érték füzér képét adja meg:

```
10 A=65
20 A$=STR$(A)
30 ? A" AZ";A$;" ERTEKE"
```

A négyzetgyök függvény (SQR)

50 négyzetgyökének kiszámításához írja be a következő programot:

? SQR(50) **RETURN**

A fenti módon bármely pozitív szám négyzetgyökét meg lehet határozni.

Az abszolútérték függvény (ABS)

Az ABS függvény igen hasznos, ha negatív számokkal dolgozunk. Ezzel a függvénnyel megkaphatjuk bármely (pozitív vagy negatív) szám pozitív értékét. Írja be a következő példákat:

? ABS(−10) **RETURN**
? ABS(5)"ANNYI MINT"ABS(−5) **RETURN**

A STOP ÉS A CONT (CONTINUE = FOLYTATNI) PARANCS

A géppel megállíttathatjuk a programot, és ha készen vagyunk, újra elkezdhetjük futtatni. A STOP parancsnak benne kell lennie a programban, az utasítást bárhol el lehet helyezni a program során. Ha a gép szünetelteti a programot, direkt módban megadott parancsokkal pontosan megtudhatjuk, mi is megy végbe valójában a programban. Pl. megtudhatjuk a ciklusváltozó vagy más változó értékét, tehát a program nyomonkövetésének, a hibakeresésnek, ill. a módosításnak hatékony eszköze. Törölje a gép tárát és írja be a következő programot:

```
10 X=INT(SQR(630))
20 Y=(.025*80)↑2
30 Z=INT(X*Y)
40 STOP
50 FOR J=0 TO Z STEP Y
60 ? "STOP ES CONTINUE"
70 NEXT
80 END
```

Most futtassa a programot. A képernyőn megjelenik a BREAK IN 40 (szünetelek a 40-es sorban) üzenet. Itt a számítógép már kiszámította X,Y és Z értékeit. Ha meg akarja tudni, mit kell csinálnia a gépnek a program további részében, írassa ki X,Y és Z értékeit (PRINT X;Y;Z). Gyakran előfordulhat egy hosszú program vagy egy rövidebb, de bonyolultabb program során, hogy egy bizonyos ponton kíváncsiak vagyunk egy változó értékére.

Ha már megkapta a szükséges információt, írja be a CONT (CONTinue) parancsot, és nyomja meg a RETURN-t, feltéve, hogy nem írt semmit a képernyőre. A számítógép ekkor folytatja a programot a STOP parancsot követő sortól.

****

Ez, valamint az előző alfejezet a BASIC programnyelvet és lehetőségeit ismertette. A fejezet hátralevő négy alfejezete azokat a parancsokat tartalmazza, amelyek kizárólag Commodore 128-as üzemmódban használhatók. Ezek közül néhány a C64-es üzemmódban egyáltalán nem működik, a többi C128-as paranccsal ugyanazokat a műveleteket lehet elvégezni, mint bizonyos C64-es parancsokkal, csak egyszerűbben. A könyvben szereplő minden parancsról vagy programozási fogásról bővebbet a C128-as Programozói kézikönyvben található. Valamennyi Commodore 7.0 parancs szintaxisát az V. fejezetben levő BASIC 7.0 kislexikon tartalmazza.

5. ALFEJEZET

Néhány, kizárólag C128-as üzemmódban használatos BASIC parancs és billentyűművelet

Bevezetés

Bonyolultabb ciklusok
- A DO/LOOP utasítás 46
- Az ELSE záradék és az IF...THEN együttes használata 46
- A BEGIN/BEND utasításpár és az IF...THEN együttes használata 47
- A SLEEP parancs 47

A kiírás formátumának meghatározása
- A PRINT USING parancs 47
- A PUDEF parancs 48

Mintaprogram

Adatbevitel a GETKEY paranccsal

A programozást megkönnyítő parancsok
- Programok bevitele 49
- A programon belüli problémák felismerése

Ablakhasználat
- A WINDOW parancs használata ablak készítésére 51
- Ablak készítése az ESC billentyűvel 51

A 2 MHz-es működési mód 52
- A FAST (gyors) és a SLOW (lassú) parancs ....

Kizárólag C128-as módban használható billentyűk
- Funkcióbillentyűk 52
- A funkcióbillentyűk átprogramozása 53
- Egyéb, csak a C128-s módban használatos billentyűk . 53

BEVEZETÉS

Ebben az alfejezetben olyan hasznos BASIC parancsokról és utasításokról lesz szó, amelyekkel eddig valószínűleg még gyakorlott BASIC programozó létére sem találkozott. Ha járatos a BASIC programozásban, biztos sokszor került már olyan helyzetbe, amikor jó hasznát vette volna az itt következő parancsoknak és utasításoknak. A könyvnek ez a része kifejti az egyes parancsok fogalmát, és példákkal világítja meg használatukat. (Teljes felsorolásuk és magyarázatuk az V. fejezetben levő BASIC 7.0 kislexikonban található.) Ugyancsak ez az alfejezet ismerteti a C128-as üzemmódban alkalmazható különleges billentyűk használatát is.

BONYOLULTABB CIKLUSOK

A DO/LOOP utasítás

Ezzel az utasítással a GOTO, GOSUB, FOR... NEXT utasításnál kifinomultabb, fejlettebb módon lehet ciklusokat előállítani. Ez az utasításkombináció olyan hasznos és sokoldalú eljárás, amely általában csak a strukturált programnyelvekre jellemző. A lehetőségek közül most csak néhányat sorolunk fel.

Ha végtelen ciklust akar előállítani, kezdje a DO utasítással, majd írja be azt a sort vagy sorokat, amelyek a végrehajtandó utasításokat tartalmazzák. Zárja a ciklust a LOOP utasítással, pl.:

```
100 DO
110 PRINT "ISMETLES"
120 LOOP
```

A programot csak a RUN/SOP billentyűvel lehet leállítani.
A DO sort követő utasításokat addig hajtja végre a gép, amíg el nem éri a LOOP sort (itt a 120-as sor), majd visszamegy a DO-ra. Így, bármi áll a DO és a LOOP között, végtelen sokszor ismétlődik.
Until Igen hasznos a DO/LOOP-ot az UNTIL (mindaddig, amíg...nem...) utasítással kombinálni. Az UNTIL a ciklus irányításának feltételét szabja meg. A ciklus vég nélkül fut mindaddig, amíg az UNTIL-ban foglalt feltétel nem teljesül.

```
100 DO: INPUT "SZERETED
A SZAMITOGEPET?";A$
110 LOOP UNTIL A$ = "IGEN"
120 PRINT "KOSZONOM"
```

A DO/LOOP utasítást igen gyakran arra használjuk, hogy egy programon belül végtelen ideig ismételjen egy teljes rutint, mint a következő példában:

```
10 PRINT "A PROGRAM MINDADDIG FUT,
   AMIG BE NEM IRJA "ELEG""
20 DO UNTIL A$ = "ELEG"
30 INPUT "FOK FAHRENHEIT";F
40 C=(5/9)*(F-32)
50 PRINT F;" FOK FAHRENHEIT ANNYI
   MINT";C; " FOK CELSIUS"
60 INPUT "MEG EGYSZER VAGY ELEG";A$
70 LOOP
80 END
```

A DO/LOOP-ot számlálóként is használhatjuk, ahol az UNTIL utasítás az ismétlések számának meghatározására szolgál.

```
10 N=2*2
20 PRINT " KETTO MEGDUPLAZVA
EGYENLO";N
30 DO UNTIL X=25
40 X=X+1
50 N=N*2
60 PRINT "MEGDUPLAZVA";
X+1;"-SZOR...";N
70 LOOP
80 END
```

Figyelje meg, hogy ha az ellenparancsot kihagyja (a 30-as sor UNTIL X=25 részét), a szám vég nélkül duplázódik, míg meg nem jelenik az OVERFLOW hibaüzenet.
While A WHILE (mialatt) utasítás az UNTIL-hoz hasonlóan működik, de a ciklus csak az alatt az időtartam alatt ismétlődik, amíg a feltétel érvényben van, mint a következő programban:

```
100 DO: INPUT "SZERET
PROGRAMOZNI?";A$
110 LOOP WHILE A$ "IGEN"
120 PRINT "KOSZONOM"
```

Exit Az EXIT (kijárat) utasítást be lehet építeni a DO/LOOP ciklusba. Ha a gép találkozik az EXIT-tel, a program a LOOP utasítást követő sorba ugrik.

Az ELSE záradék és az IF...THEN együttes használata

Az ELSE (különben, egyébként) záradékkal közölhetjük a számítógéppel, hogy mit tegyen, ha az IF...THEN utasítás feltétele nem igaz. Mielőtt a gép a következő sort folytatná, végrehajta a parancsot vagy az ELSE sorba ugrik. Pl.: ha azt akarjuk, hogy a számítógép kiírja egy szám négyzetét, a következőképpen használjuk az ELSE-t:

```
10 INPUT "IRJA BE A SZAMOT, AMELYET
NEGYZETRE AKAR EMELNI";N
20 IF N<100 THEN PRINT N*N : ELSE 40
30 END
40 ?"A SZAMNAK SZAZNAL KISEBBNEK
KELL LENNIE": GOTO 10
```

Az IF...THEN és az ELSE közé kettőspontot kell tenni.

A BEGIN/BEND utasításpár és az IF...THEN együttes használata

A BASIC 7.0 az IF...THEN feltételt egy lépéssel továbbfejlesztette. A BEGIN/BEND utasításpár használatával, abban az esetben, ha az IF feltétel igaz, nemcsak egy műveletet vagy a GOTO utasítást lehet elvégezni, de több programsort is. A parancs szerkezete a következő:

```
IF feltétel THEN BEGIN:
(programsorok):
BEND:ELSE
```

Ne felejtsen el a BEGIN és a végrehajtandó utasítások, valamint a közbezárt utolsó parancs és a BEND szó közé kettőspontot tenni. A BEGIN/BEND-et lehet ELSE nélkül is használni, vagy az ELSE-t követően is, ha a THEN után csupán egyetlen parancs áll. Próbálja ki ezt a programot:

```
10 INPUT A
20 IF A < 100 THEN BEGIN: ? "AZ ON
SZAMA ";A;" VOLT"
30 SLEEP 2:REM KESLELTETES
40 FOR X=1 TO A
50 ? "EZ EGY PELDA A BEGIN/BEND
UTASITASRA"
60 NEXT X
70 ?"ELEG":BEND:ELSE ?"TUL SOK"
80 END
```

Ez a program egy számot kér a használótól. Ha a szám kisebb, mint 100, a gép a BEGIN és a BEND közötti utasításokat hajtja végre, valamint a BEND-del egy sorban levő valamennyi utasítást, az ELSE kivételével. A képernyőn megjelenik az AZ ON SZAMA N üzenet. A 30-as sor késleltető ciklus, amely hosszabb ideig hagyja fent az üzenetet, hogy megkönnyítse az elolvasását. Ezután egy FOR...NEXT ciklus következik, hogy a használó által megadott n-szer kiírja az üzenetet. Ha a szám nagyobb, mint 100, a THEN feltételt a program átugorja és az ELSE feltételt (a "TUL SOK" kiírást) teljesiti. Az ELSE kulcsszónak egy sorban kell állnia a BEND-del.

A SLEEP parancs

Nézzük meg a most tárgyalt program 30-as sorában levő SLEEP parancs működését. A SLEEP paranccsal lényegesen egyszerűbben lehet a programba késleltetéseket beiktatni.
A SLEEP parancs formátuma:

```
SLEEP n
```

amelyben az n a késleltetés idejét jelenti másodpercben. Az n-re 1 és 65 535 közötti értékeket lehet megadni; a 30-as sor parancsában levő 2-es szám 2 másodpercnyi késleltetést jelent.

A KIÍRÁS FORMÁTUMÁNAK MEGHATÁROZÁSA

A PRINT USING parancs

Tegyük fel, hogy olyan programot írunk, amellyel egy kereskedő a napi dollárbevételét számítja ki. Ha az eladások összértékét elosztjuk a vásárlók számával, akkor megkapjuk az egy vásárlóra jutó, átlagos vásárlási összeget. Előfordulhat, hogy az eredmény kijelzéséhez négy vagy öt decimális helyiértékre lesz szükség. A számítógépet utasíthatjuk arra, hogy az eredményt az általunk meghatározott formátumban írja ki. Erre szolgál a PRINT USING parancs.
A PRINT USING paranccsal meghatározható a kiírás formátuma, amelyben szóközöket, vesszőket, tizedespontokat és $ jeleket helyezhetünk el. A # jel (hash mark) a kijelzésre kerülő eredményben a szóközök vagy a karakterek helyét jelöli. Pl. a

```
PRINT USING "#$#####.##"; A
```

arra utasítja a számítógépet, hogy az A változót a PRINT USING parancsban megadott formátum szerint, tehát a tizedesponttól balra öt helyre, attól jobbra pedig két helyre írja ki. A $ jelet megelőző # jel arra utasítja a gépet, hogy a $ jel a kiírandó formátumban mindig az első karakter előtt álljon.
Ha azt akarjuk, hogy pl. három számjegyet kijelző karakterhely előtt egy vessző jelenjen meg – tehát a kijelzés formátuma $1,000.00 legyen –, akkor ezt szintén a PRINT USING utasításban kell megadni. A kiírásban ezzel az utasítással meghatározhatunk szóközöket, vesszőket, tizedespontokat és $ jeleket. Ezen túlmenően más karakterek is megadhatók a PRINT USING utasításban (l. a BASIC kislexikont).

A PUDEF parancs

Ha a megformált kijelzésben nem dollárt és centet használ, írja be a PUDEF (Print Using DEFine) parancsot. A négy formátumkarakter bármelyikét helyettesítheti bármelyik billentyűvel.
A PUDEF parancsnak négy pozíciója van, de nem kell mind a négyet újra meghatározni. A parancs a következő:

```
PUDEF" _ , . $ "
       1 2 3 4
```

ahol
- az 1. pozíció a kitöltő (helypótló) karakter. Ha nem határozunk meg új pozíciót, üres karakterhely fog megjelenni;
- a 2. pozíció a vessző karakter. Ez az alapértelmezés;
- a 3. pozíció a tizedespont;
- a 4. pozíció a dollárjel.

Ha olyan programot ír pl., amely dollárösszegeket vált át angol fontra, a kiírás a következő parancsokkal formálható:

```
10 PUDEF "£"
20 PRINT USING "#$####.##";X
```

MINTAPROGRAM

Ez a program kamat- és kölcsönkifizetéseket számol az imént tanult parancsok és utasítások felhasználásával. A kölcsönnek minimum értéket ad az IF...THEN utasítással és az ELSE záradékkal, majd a PRINT USING paranccsal meghatározza a formátumot.

```
10 INPUT " A KOLCSON OSSZEGE
DOLLARBAN";A
20 IF A<100 THEN 80: ELSE P=.15
30 I=A*P
40 ?"A KIFIZETES OSSZEGE";
50 PRINT USING "#$#####.##";A+I
60 GOTO 80
70 ?"100$ ALATTI KOLCSON NEM VEHETO
FEL"
80 END
```

ADATBEVITEL A GETKEY PARANCCSAL

Azt már láttuk, hogyan lehet az INPUT és a GET parancsokkal adatot beírni a programba. Egy éppen futó programba a GETKEY utasítással is el lehet érni ugyanezt. A GETKEY utasítás egy időben csak egy billentyűt fogad el, és általában egy füzérváltozó követi, pl. A$. A gép bármelyik lenyomott billentyűt beolvassa ebbe a füzérváltozóba. A GETKEY azért igen hasznos, mert úgy lehet vele – egy időben csak egy – karaktert beírni, hogy nem kell minden egyes esetben külön újra megnyomni a karakterek után a RETURN billentyűt. A GETKEY utasítás kizárólag programban alkalmazható. Egy példa a GETKEY használatára:

```
1000 PRINT "VALASSZON A, B, C, D, E ES
F KOZUL"
1010 GETKEY A$
1020 PRINT A$; "-T VALASZTOTTA"
```

A gép vár, amíg meg nem nyomunk egy billentyűt. Amikor megnyomtuk, a karaktert az A$ változóhoz rendeli, majd kiírja a 1020-as sorba.
A következő program a GETKEY-t egy összetettebb és hasznosabb alkalmazásban mutatja be, és egy többesélyes kérdés megválaszolásánál használhatjuk. A számítógép azt is megkérdezi, hogy meg kell-e ismételnie a kérdést. Ha az adott válasz helytelen, a használó újra próbálkozhat, az I billentyű lenyomása után (80-as sor). A válaszul megnyomott billentyű az A$ változóba kerül, míg a MEGPROBALOD UJRA? üzenetre adott válasz a B$-ba, a 60-as és a 90-es sor GETKEY utasításai alapján. A program ciklusaihoz az IF...THEN utasításokat használjuk, hogy a gép a különböző billentyűbevitelekre megfelelően reagáljon.

```
10 PRINT "MELYIK KET BOLYGONAK NINCS
HOLDJA?"
20 PRINT "A. MERKUR ES JUPITER"
30 PRINT "B. URANUSZ és NEPTUNUSZ
40 PRINT "C. MERKUR ES VENUSZ
50 PRINT "D. JUPITER ES MARS"
60 GETKEY A$
70 IF A$= "C" THEN 150
80 PRINT "NEM JO A VALASZ.
MEGPROBALOD UJRA? (I VAGY N)"
90 GETKEY B$
100 IF B$="I" THEN PRINT "A,B,C VAGY
D?":GOTO 60
110 IF B$="N" THEN 140
120 PRINT "CSAK I-VEL VAGY N-NEL
FELELHETSZ!"
130 GOTO 90
140 PRINT "A HELYES VALASZ C"
145 GOTO 160
150 PRINT "GRATULALOK!"
160 END
```

A GETKEY sokban hasonlít a GET-hez, csak a GETKEY automatikusan várja, hogy egy billentyűt lenyomjunk.

A PROGRAMOZÁST MEGKÖNNYÍTŐ PARANCSOK

A korábbi fejezetekből megtanulhatta, hogyan lehet egy programon változtatni, és a hibákat kijavítani az INST/DEL parancs segítségével. A BASIC nyelvben vannak más parancsok és függvények is, amelyek segítenek meghatározni a tényleges programozásbeli hibák helyét, valamint olyan parancsok, amelyek használata kényelmesebbé, egyszerűbbé teszi a programozást.

Programok bevitele

AUTO A C128-asnak önműködő számlálója van. Határozzuk meg a sorszámok növekedését, mondjuk legyen a szokásos 10. Még a program megkezdése előtt írja be direkt módban a következőt:

AUTO 10 **RETURN**

A gép automatikusan, tízesével beszámozza a programot. Amikor megnyomja a RETURN-t, megjelenik a következő sorszám és a kurzor is a megfelelő helyen lesz, tehát beírhatja a következő utasítást. A számsor tetszőleges számmal növelhető, 5-tel, de akár 50-nel is. Csak írja be a számot az AUTO szó után, majd nyomja meg a RETURN billentyűt. Ha a számozást meg akarja szüntetni, írja be az AUTO parancsot, de növekedés nélkül, és nyomja meg a RETURN-t.

RENUMBER Ha ír egy programot, és később még hozzá akar tenni utasításokat, előfordulhat, hogy nehézkes megoldani a sorszámozást. A RENUMBER (újraszámozás) paranccsal a sorszámokat azonos növekményű számokká lehet átalakitani a program egészében, vagy egy bizonyos részében. A RENUMBER parancsnak opcionális paraméterei vannak, l. alább, zárójelben:

RENUMBER [új kezdősor[növekmény[,régi kezdősor]]]

Az új kezdősor a beszámozás utáni első programsor száma lesz. Ha külön nem adja meg, az alapérték 10. Ugyancsak 10 a növekedés is. A régi kezdősor száma az a sorszám, ahol az átszámozásnak el kell kezdődnie. Ezzel az eljárással inkább a program egy részét lehet átprogramozni, mint magát az egész programot. Alaphelyzet a program első sora. Pl. a

RENUMBER 40,,80

parancs azt közli a számítógéppel, hogy a 80-as sortól kezdve számozza át a programot, 10-esével. A 80-as sor lesz a 40-es sor. Ezt a parancsot is, mint az AUTO-t, csak direkt módban lehet használni.

DELETE Azt már tudja, hogy a sorszám beírásával és a RETURN billentyű lenyomásával hogyan lehet sorokat törölni, a módszer azonban nehézkes, ha a program egy nagyobb részéről van szó. A DELETE (töröljl) paranccsal időt takarít meg, mert egyszerre több sort is lehet ily módon törölni. Pl. a

DELETE 10–50

parancs a 10-es és az 50-es, valamint minden közöttük levő sort töröl. A DELETE használata hasonló a LIST parancséhoz, amennyiben az elvégzendő feladat helyét meg lehet vele határozni egy bizonyos sorig, egy bizonyos sortól kezdve, vagy csak egy bizonyos sorban. Pl.:

DELETE –120	minden sort töröl, a 120-asig bezárólag,
DELETE 120–	törli a 120-as sort és mindent, ami utána következik,
DELETE 120	csak a 120-as sort törli.

A PROGRAMON BELÜLI PROBLÉMÁK FELISMERÉSE

Ha egy program nem a várt módon működik, általában egy hibaüzenet jelenik meg. Néha azonban a hibaüzenetekből sem derül ki a probléma lényege. A Commodore 128-as számítógépen több lehetőség is van a hiba helyének megállapítására.

HELP A HELP parancs megadja a hibás sor számát, ehhez mindössze a főbillentyűzet fölött található HELP billentyűt kell lenyomni. Írja be a következő utasítást. Szándékosan van benne hiba, tehát pontosan így írja be:

10 ?3;4:5;6

Amikor ezt az egysoros programot futtatja, a gép rendben kiírja a 3-at és a 4-et, azután azonban megjelenik a SYNTAX ERROR IN 10 (szintaktikai hiba a 10-es sorban) hibaüzenet. Tegyük fel, hogy nem látja a hibát (kettőspont van pontosvessző helyett a 4 és az 5 között). Ha megnyomja a HELP billentyűt vagy beírja a HELP **RETURN** parancsot, a sor ismét megjelenik, de az 5 és a 6 erősebb fénnyel világít, hogy megmutassa, a hiba ott van.

Hibakeresés – a TRAP parancs

Általában, ha a programban valahol hiba van, a program leáll. Ekkor le kell ütni a HELP billentyűt, hogy ki lehessen nyomozni a hibát. A BASIC 7.0 TRAP paranccsal azonban magába a programba is be lehet építeni a hibakereső képességet. A TRAP parancs megkönnyíti a hiba helyének megállapítását és kijavítását, majd folytatja a program futtatását. A hibakeresőt általában a program első sorába teszszük. A

```
5 TRAP 100
```

parancs azt közli a számítógéppel, hogy ha hibát talál, menjen a megadott sorba (ez jelen esetben a 100-as sor). A 100-as sor megjelenik a program végén, fenntartva az előre nem látott lehetőségek számára. Ha nincsen hiba a programban, a két sor közül egyiket sem hajtja végre a gép. Ha hiba fordul elő, a TRAP utasítást tartalmazó sor életbe lép, és a vezérlés a program egy másik részére irányul. Ezeket az utasításokat előre feltételezhető hibák kijavítására használjuk. Ilyenek többek között adatbevitelnél vagy grafikus módból szöveg módba való visszatérés során keletkezhetnek. Ha a DO/LOOP mintaprogramot (megduplázott számok) lefuttatja, és nem írt bele UNTIL utasítást, előfordulhat, hogy OVERFLOW hibaüzenet jelenik meg és a program leáll. Ezt meg lehet előzni úgy, hogy a programhoz az elején és a végén hozzáadunk 1–1 sort, pl. a következőket:

```
5 TRAP 100
100 IF N>1 THEN END
```

Bár az N a program folyamán végig sokkal nagyobb volt, mint 100, az utasítás mindaddig nem érvényes, amíg hiba nem fordul elő. Amikor a szám „túlcsordul" (overflow), azaz nagyobb, mint amit a számítógép be tud fogadni, életbe lép a TRAP utasítás. Mivel N nagyobb, mint 1, a program *előbb végetér, mintsem hibaüzenettel* félbeszakadjon.

Egy olyan példa következik, ahol a hibakeresést arra használjuk fel, hogy a 0 számot ne lehessen bevinni osztónak:

```
10 TRAP 1000
100 INPUT "BARMILYEN
    SZAMMAL TUDOK OSZTANI.
    KEREM AZ OSZTANDOT";D
110 INPUT "MI AZ OSZTO?";B
120 A=D/B
130 PRINT D;"OSZTVA";B;"ANNYI
    MINT";A
140 END
1000 IF B=0 THEN PRINT"ERRE
     MEG EN SEM VAGYOK KEPES"
1100 INPUT "MAS SZAMOT
     KEREK";B:RESUME 120
```

Bizonyára észrevette az 1100-as sorban a RESUME-ot. Ez azt közli a géppel, hogy menjen vissza a jelzett sorba (ebben az esetben a 120-as) és folytassa a program végrehajtását. A folytatás a megtalált hiba természetétől függően lehetséges vagy nem. A hibakeresésről további információt nyújtanak az ERR$, az EL és az ER parancsok, amelyek az V. fejezet BASIC 7.0 kislexikonban találhatók.

Programok nyomkövetése – a TRON és a TROFF parancs

Ha a programban valahol probléma van, vagy nem a várt eredményt kapja, hasznos lehet, ha módszeresen végigmegy a programon, és lépésről lépésre végigcsinálja azt, amit a számítógép is tenne. Ezt a folyamatot nyomkövetésnek hívják. Csináljon változórekeszt és aktualizálja (írja felül) az értékeket az utasításnak megfelelően. Végezze el a számításokat, és minden egyes utasítás után írja ki az eredményt. Egy ilyen folyamat során kiderülhet pl., hogy a GOTO-t helytelen sorszámmal használta, vagy kiszámított ugyan egy eredményt, de nem olvasta be a változóba. Sok programozási hibát lokalizálni lehet, ha úgy teszünk, mint a gép tenne és egyszerre csak egy utasítást követünk. A Commodore 128-as számítógép a TRON (TRace ON) és a TROFF (TRace OFF) különleges parancsokkal képes a nyomkövetésre. Amikor a program fut, a TRACE ON segítségével a gép a végrehajtás sorrendjében írja ki a sorszámokat, valamint az eredményeket is. Így kiderül, miért nem kaptuk meg a várt eredményt.
Írja be bármelyik eddig használt rövid programot vagy írjon egy újat. Direkt módban gépelje be a TRON parancsot. Ha futtatja a programot, figyelje

meg, hogy még az eredmények előtt megjelennek a sorszámok zárójelben. Kövesse figyelemmel a sorszámokat és nézze meg, hány lépésre volt szüksége a gépnek, hogy egy bizonyos ponthoz elérjen. A TRON érdekesebb, ha egy sokleágazású, pl. GOTO, GOSUB és IF...THEN sorokat tartalmazó programot választ. A TROFF parancs beírásával lehet kikapcsolni a nyomkövető módot, mielőtt továbbfuttatná a programot. Nem szükséges az egész programot nyomonkövetni, a program problémás része elé is be lehet tenni a TRON utasítást, utána pedig a TROFF sort. Futtatáskor következésképpen, csak a TRON és a TROFF közé eső sorok jelennek meg zárójelben.

ABLAKHASZNÁLAT

Az ablakok a képernyőnek azon külön területei, amelyeket a munka számára kijelölünk. Bármit, amit beírunk (sorok, programlisták stb.), az ablak definiálása után, az ablak keretei közt jelenik meg, az ablak területén kívül eső részek érintetlenül maradnak. A Commodore 128-as számítógéppel kétféleképpen lehet ablakot készíteni: a WINDOW paranccsal és az ESCAPE billentyű funkcióival.

A WINDOW parancs használata ablak készítésére

A Commodore 128 BASIC 7.0 nyelvben létezik egy parancs, a WINDOW, amely lehetőséget nyújt ablakok készítésére és manipulálására. A parancs formátuma a következő:

WINDOW bal felső oszlop, bal felső sor, jobb alsó oszlop, jobb alsó sor [, opcionális törlés]

A WINDOW utáni első két szám a leendő ablak bal felső sarkának oszlop- és sorszámát jelöli, a következő kettő pedig a jobb alsó sarokét. Ne feledje, hogy e koordináták tartományait a képernyőformátum (40 vagy 80 oszlop) szabja meg. A parancshoz opcionális tartalomtörlési lehetőséget is hozzá lehet adni. Ha a parancs végére 1-est ír, az ablak tartalma törlődik a képernyőről, pl.:

WINDOW 10, 10, 20, 20, 1

A következő mintaprogram 4 ablakot szerkeszt a képernyőre, 40 vagy 80 oszlopos formátumban.

Ablak készítése az ESC billentyűvel

Ha az ESC (Escape) billentyűvel akar ablakot készíteni, kövesse a következő lépéseket:

1. Vigye a kurzort a leendő ablak bal felső sarkába.
2. Nyomja le, majd engedje el az ESC billentyűt és nyomja meg a T-t.
3. Vigye a kurzort a leendő ablak jobb alsó sarkába.
4. Nyomja le, majd engedje el az ESC billentyűt és nyomja le a B-t.

Kész az ablak.

Az ablakot és tartalmát az ESC billentyűvel lehet kezelni. Az ESC és egy másik billentyűvel képernyőszerkesztési feladatokat hajthatunk végre (szöveg beszúrása és törlése, scroll (képernyőgörgetés), az ablak méretének megváltoztatása stb.). Nyomja meg az ESC billentyűt és engedje el, majd nyomjon meg egy másik billentyűt a következő listából, a kívánt művelet végrehajtására:

```
10 REM **** P.2 ****
20 PRINT"█"  :REM A KEPERNYO TORLESE
30 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
40 B$=A$+A$+A$
50 FOR I=1 TO 25:PRINTB$:NEXT:REM  KEPERNYO TOLTESE KARAKTEREKKEL
60 WINDOW 1 ,1 ,8 ,20 :REM ELSO ABLAK BEALLITASA
70 PRINT"█
80 REM AZ ELSO ABLAK TOLTESE PIROSSAL
90 WINDOW 15,15,39,20,1 :REM MASODIK ABLAK BEALLITASA
100 PRINT"█"; B$;A$            :REM ABLAK TOLTESE KARAKTEREKKEL
110 WINDOW 30,1,39,22,1 :REM HARMADIK ABLAK BEALLITASA
120 PRINT"█": LIST              :REM SARGA KIVALASZTASA ES LISTA AZ ABLAKBA
130 WINDOW 5,5,33,18,1  :REM NEGYEDIK ABLAK BEALLITASA,HASONLOAN
A HARMADIKHOZ
140 PRINT"█":PRINTA$:LIST: REM SZIN VALTOZTATAS - A$ KIIRASA ES LISTA
AZ ABLAKBA
```

@ mindent töröl a kurzortól az ablak végéig;
A automatikus beszúrás;
B az ablak jobb alsó sarkát a kurzor pillanatnyi helyére teszi;
C törli a beszúró és idéző módot;
D törli az aktuális sort;
E a kurzor nem villog;
F a kurzor villog;
G csengő bekapcsolása (a Control G segítségével);
H csengő kikapcsolása;
I sor beszúrása;
J az aktuális sor elejére megy vissza;
K az aktuális sor végére megy;
L scroll bekapcsoló;
M scroll kikapcsoló;
N vissza a normál (nem inverz) kijelzéshez (csak 80 oszlop esetében);
O törli az automata beszúrási lehetőséget;
P mindent töröl a sor elejétől a kurzorig;
Q mindent töröl a kurzortól a sor végéig;
R inverz képernyőkijelzés (csak 80 oszlop esetén);
S váltás tele kurzorra (■);
T az ablak bal felső sarkát a kurzor pillanatnyi helyére teszi;
U váltás aláhúzó kurzorra (_);
V scroll (képernyőgörgetés) egy sort felfelé;
W scroll (képernyőgörgetés) egy sort lefelé;
X átkapcsolás 40 és 80 oszlop között;
Y visszaállítja a TAB megállókat (stop) alaphelyzetbe;
Z törli az összes TAB megállót.

Kisérletezzen az ESC billentyűvel. Bizonyára egyes funkciókat hasznosabbaknak talál, másokat kevésbé. Ne feledje, hogy a szokásos INST/DEL billentyűt is használhatja ablakon belüli szövegszerkesztésre.

Ha az ablak kész, minden képernyőkijelzés a meghatározott „dobozba" (keretbe) kerül. Ha törölni kívánja az ablak tartalmát, nyomja meg a SHIFT és a CLEAR/HOME billentyűket együtt, mire eltűnik az ablak, a kurzor pedig a képernyő bal felső sarkában jelenik meg. Az ablakhasználat igen hasznos programok írásakor, listázásakor és futtatásakor, mert lehetővé teszi, hogy a képernyő egy bizonyos részén dolgozzunk, míg a többi rész változatlanul marad.

A 2 MHz-ES MŰKÖDÉSI MÓD

A FAST (gyors) és a SLOW (lassú) parancs

A 2 MHz-es működési módban lehetősége van, hogy nem grafikus programokat 80 oszlopos formátumban a normál sebesség kétszeresével futtasson. Erre szolgál a FAST és a SLOW parancs. A FAST parancs a számítógépet a 2 MHz-es módra váltja át. Formátuma a következő:

FAST

A SLOW parancs visszaállítja az 1 MHz-es módot. A parancs formátuma:

SLOW

KIZÁRÓLAG C128-AS MÓDBAN HASZNÁLHATÓ BILLENTYŰK

Funkcióbillentyűk

A numerikus billentyűzet fölötti négy billentyű különleges funkciókat lát el, használatukkal időt lehet megtakarítani, mert egyetlen leütéssel lehetővé teszik ismétlődő feladatok elvégzését. Az első felirata F1/F2, a másodiké F3/F4, a harmadiké F5/F6, a negyediké F7/F8. Az 1–4 funkciókhoz a billentyűt egymagában kell lenyomni, míg az 5–8 funkciókhoz a SHIFT-tel együtt.
A billentyűk szabványfunkciói:

F1	F2	F3	F4
GRAPHIC	DLOAD"	DIRECTORY	SCNCLR
F5	F6	F7	F8
DSAVE"	RUN	LIST	MONITOR

Részletezve:

Az 1. billentyű átvált valamelyik grafikus módba, ha megadjuk a grafikai terület számát és megnyomjuk a RETURN-t. A CIRCLE és a PAINT grafikai parancsoknál használjuk. Bővebbet a 6. alfejezetben.

A 2. billentyű kiírja a DLOAD" üzenetet a képernyőre. Mindössze a programnevet kell beírnia, zárni az idézőjelet, megnyomni a RETURN billentyűt és máris lehet tölteni programot lemezről, a DLOAD sajátkezű begépelése nélkül.

A 3. billentyű listázza a lemezegységben található lemez tartalomjegyzékét.

A 4. billentyű törli a képernyőt a SCNCLR paranccsal.

Az 5. billentyű kiírja a DSAVE" üzenetet a kép-

ernyőre. Csak a programnevet kell beírnia, megnyomni a RETURN-t és a programot máris ki lehet menteni lemezre.

A 6. billentyű ezzel lehet a mindenkori programot futtatni.

A 7. billentyű kilistázza az éppen futó programot.

A 8. billentyű gépi nyelv monitorra vált (l. J) függelék).

A funkcióbillentyűk átprogramozása

A funkcióbillentyűk szükség szerint újraprogramozhatók. A KEY parancs használatával ez a folyamat igen egyszerű. BASIC programból vagy akár direkt módban is bármikor újra lehet programozni a billentyűket. Erre akkor lehet szükség, ha gyakran használ egy parancsot és időt akar megtakarítani a parancs ismételt begépelésének elkerülésével. Ha a gépet kikapcsolja, az új funkciók megszűnnek. Tetszőleges számú billentyűt bármennyiszer át lehet programozni. Ha pl. az F7 billentyűt akarja újraprogramozni, hogy az igényes kivitelű vagy színes grafikus módból visszatérjen szöveg módba, a következők szerint használja a parancsot:

```
KEY 7,"GRAPHIC 0" + CHR$(13)
```

A CHR$ (13) a RETURN ASCII karakterkódja. Így, ha az átprogramozás után megnyomja az F7 billentyűt, a GRAPHIC 0 parancs automatikusan kiíródik és a RETURN beviszi a gépbe. Teljes parancsokat vagy parancssorozatokat is lehet egyetlen billentyűhöz rendelni.

Egyéb, csak a C128-as módban használatos billentyűk

HELP Mint arról már korábban szó esett, ha a programba hiba került, megjelenik a hibaüzenet. Az A) függelékben ezekről bővebben esik szó. A HELP billentyű a hibák azonosításában van segítségére. A hibaüzenet megjelenése után nyomja meg a HELP billentyűt. Ekkor a hibás sor inverzben, nagyobb fényerővel (40 oszlop esetén), vagy aláhúzva (80 oszlop esetén) jelenik meg. Pl.:

?SYNTAX ERROR IN LINE 10 – ez jelenik meg a képernyőn

HELP – megnyomja a HELP billentyűt

10 PRONT "COMMODORE SZAMITOGEP"

A hiba inverzben, nagyobb fényerővel jelenik meg 40 oszlopos kijelzés esetén, vagy aláhúzva 80 oszlopos kijelzés esetén.

NO SCROLL A billentyű lenyomásával a szöveg megáll, ha a kurzor eléri a képernyő alját; mindaddig áll, míg újra le nem nyomjuk a billentyűt.

CAPS LOCK Ezzel a billentyűvel a SHIFT használata nélkül is csupa nagybetűt lehet írni. A billentyű nyomásra zár, és újra le kell ütni, ha nyitni akarjuk. A CAPS LOCK kizárólag a betűvel ellátott billentyűkre vonatkozik.

40/80 kijelző A 40/80 billentyűvel határozhatjuk meg a képernyő alapformátumát, ez lehet 40 vagy 80 oszlopos. Ezt a billentyűt kizárólag a számítógép bekapcsolása vagy alaphelyzetbe való visszaállítása előtt lehet használni. Bekapcsolás után ezzel a billentyűvel nem lehet üzemmódot váltani. (A 8. alfejezetben találja a 40/80 oszlopos üzemmód magyarázatát.)

ALT Az ALT billentyűvel egy bizonyos billentyűnek vagy billentyűsornak speciális jelentést adhat. Külön alkalmazási program nélkül az ALT és valamelyik másik billentyű lenyomása hatástalan marad.

TAB Ez éppúgy működik, mint az írógép tabulátor billentyűje. Tabulátor megállókat lehet vele meghatározni vagy törölni és a kurzort a TABulált (beállított) oszlopokhoz vinni.

LINE FEED Hasonlóan a „kurzor le" billentyűhöz, a LINE FEED (soremelés) megnyomásával a kurzort a következő sorba lehet vinni.

* *

Ebben az alfejezetben mindössze néhány olyan fogalmat, billentyűt és parancsot tárgyaltunk, amelyek a Commodore 128-as számítógépet különlegessé teszi. További magyarázatokért lapozza fel az V. fejezetben található BASIC kislexikont.

6. ALFEJEZET

A C128-asra jellemző szín-, animációs és sprite-grafikus utasítások

Bevezetés a grafikába
Grafikus sajátságok 56
A grafikus parancsok felsorolása 56

Grafikus programozás a Commodore 128-assal
Színkiválasztás 56
A képernyőkijelzés típusai 57
Váltás grafikus módba 57
Grafika megjelenítése a képernyőn 58
Karakterek megjelenítése a bittérképes képernyőn – a CHAR parancs 59
A grafikus alakzatok (formák) méretének megváltoztatása – a SCALE parancs 60
Grafikus mintaprogram készítése 60

A sprite-ok: mozgatható, programozható tárgyak
Sprite-ok készítése 62
Sprite-utasítások a programban 62
A sprite képének megrajzolása 62
A sprite adatainak tárolása a SSHAPE utasítással 63
Képi adatok átmentése a sprite-ba 63
A sprite bekapcsolása 63
A sprite mozgatása a MOVSPR utasítással 63
Sprite-program készítése 65
A sprite-definíciós mód – a SPRDEF parancs 65
Sprite készítése SPRDEF módban 65
Egymáshoz kapcsolódó sprite-ok 67
Sprite-adatok tárolása bináris file-okban 69

BEVEZETÉS A GRAFIKÁBA

C128-as módban a Commodore BASIC 7.0 nyelve sok új és jól alkalmazható parancsot és utasítást tartalmaz, amelyek megkönnyítik a grafikus programozást. A C128-as módban használható mindkét képernyőformátumot (40/80 oszlop) külön mikroprocesszor chip vezérli. A 40 oszlopos chip neve Video Interface Controller, röviden VIC. A 80 oszlopos chipre számmal utalunk, száma 8563. A VIC chip 16 színnel dolgozik és ez vezérli a nagyfelbontású grafikát, amelyet bittérképes grafikának nevezünk. A 80 oszlopos chip, amelynek színkínálata szintén 16 szín, csak karaktereket és karaktergrafikát képes megjeleníteni. Így C128-as módban a részletes grafikus programokat 40 oszlopos formátumban kell készíteni.

Grafikus sajátságok

A Commodore 128-as grafikus képességei C128-as üzemmódban:

- 12 speciális grafikus parancs;
- 16 szín;
- hat különféle kijelzési mód;
- nyolc programozható, mozgatható figura, ún. *sprite;*
- kombinált grafikus/szöveg kijelzés.

Ezek a tulajdonságok sokoldalú, könnyen kezelhető grafikus rendszert alkotnak.

A grafikus parancsok felsorolása

A grafikus parancsok rövid magyarázatát a következő felsorolásban adjuk meg:

BOX	négyszögeket rajzol a bittérképes képernyőre.
CHAR	karaktereket ír ki a bittérképes képernyőre.
CIRCLE	köröket, ellipsziseket és más geometriai idomokat rajzol.
COLOR	kiválasztja a keret, az előtér, a háttér, valamint a karakterek színét.
DRAW	vonalakat és pontokat ír a bittérképes képernyőre.
GRAPHIC	kiválasztja a képernyőkijelzést (szöveg, bittérkép vagy osztott képernyős bittérkép).
PAINT	a bittérképes képernyő egy bizonyos területét színnel kitölti.
SCALE	beállítja az alakok relatív nagyságát a bittérképes képernyőn.
SPRDEF	átvált sprite-meghatározó (definiáló) módba sprite-ok készítéséhez.
SPRITE	mozgatja, kiszínezi és beállítja a sprite képernyőprioritását és növeli a sprite-ot.
SPRSAV	szövegfüzér változót sprite tárterületen tárol és fordítva.
SSHAPE	a bittérképes képernyő egy részének képét a szövegfüzér változóban tárolja.

A legtöbb parancs használata kiderül a következő példák során. Az itt fel nem sorolt grafikus parancsok és függvények részletes magyarázata az V. fejezetben a BASIC 7.0 kislexikonban található.

GRAFIKUS PROGRAMOZÁS A COMMODORE 128-ASSAL

A következő részben lépésről lépésre haladva ismerkedünk a grafikus programozással. Az új parancsokat mindig építsük be a programba, amelyet a fejezet olvasásával párhuzamosan készítünk. Mire a fejezet végére érünk, kész lesz a komplett grafikus program is.

Színkiválasztás

Az első lépés a képernyő háttér-, előtér- és keretszínének kiválasztása. Ehhez írja be a következő parancsot:

```
COLOR forrás, szín
```

ahol a *forrás* a színezendő képernyőrészlet (háttér, előtér, keret), a *szín* pedig a hozzá tartozó színkód. A 6.1. táblázatból leolvashatjuk a forrás kódszámait, a 6.2. táblázatból a 40 oszlopos formátum színkód számait, a 6.3. táblázatból pedig a 80 oszlopos formátum színkód számait.

6.1. táblázat

Forrásszámok

Szám	Forrás
0	40 oszlopos háttérszín (VIC)
1	a grafikus képernyő előtere (VIC)
2	a színes képernyő 1. előtérszíne (VIC)
3	a színes képernyő 2. előtérszíne (VIC)
4	40 oszlopos (VIC) keret (szöveg vagy grafikus módban)
5	40/80 oszlopos szövegképernyő karakterszíne
6	80 oszlopos háttérszín (8563)

6.2. táblázat

A 40 oszlopos formátum színkód számai

Színkód	Szín	Színkód	Szín
1	fekete	9	narancs
2	fehér	10	barna
3	vörös	11	világosvörös
4	türkiz	12	sötétszürke
5	bíbor (lila)	13	középszürke
6	zöld	14	világoszöld
7	kék	15	világoskék
8	sárga	16	világosszürke

6.3. táblázat

A 80 oszlopos formátum színkód számai

Színkód	Szín	Színkód	Szín
1	fekete	9	sötétlila
2	fehér	10	sötétsárga
3	sötétvörös	11	világosvörös
4	világostürkiz	12	sötéttürkiz
5	világoslila	13	középszürke
6	sötétzöld	14	világoszöld
7	sötétkék	15	világoskék
8	világossárga	16	világosszürke

A képernyőkijelzés típusai

A Commodore 128-as számítógép különböző módokon képes információt megjeleníteni a képernyőn: a COLOR parancs „forrás" paramétere a képernyőkijelzés módjaira utal. A videokijelzés háromféle lehet.

Az első a szövegkijelzés, amely csak karakterek (betűk, számok, speciális jelek, valamint a C128-as legtöbb billentyűjének homloklapján látható grafikus karakterek) megjelenítésére képes. A Commodore 128-as mind 40, mind pedig 80 oszlopos képernyőformátumban tud szöveget megjeleníteni.
A második módot nagyfelbontású grafikához (képek s bonyolult rajzolatok) használjuk. Ezen belül van standard és színes bittérképes mód. A bittérképes mód lehetővé teszi, hogy a képernyő minden egyes pontját (*pixel* = képelem) külön vezéreljük, ezáltal meglehetősen részletes képeket és grafikákat készíthetünk. Ez csak 40 oszlopos képernyőformátum esetén lehetséges. A 80 oszlopos képernyőformátum szöveg kijelzésére alkalmas.
A szöveg és a bittérképes mód közötti különbség a képernyők információkezelésében és -tárolásában rejlik. A szövegképernyő csak teljes karaktereket képes kezelni, amelyek mindegyike 8*8 pont területet foglal el a képernyőn. A hatásosabb bittérképes mód a képernyő minden egyes pontját külön vezérli.
A harmadik, az osztott képernyőkijelzés az első kettő keveréke. Ilyenkor a képernyő egy része szöveg módban, míg a másik része bittérképes módban (standard vagy színes) dolgozik. A Commodore 128-as azért képes erre, mert a tár két külön részében tárolja a két képernyőt, az egyik rész a szövegnek, a másik a grafikának van fenntartva.

Írja be a következő rövid programot:

```
10 COLOR 0,1:REM A SZOVEG
   HATTERSZINE = FEKETE
20 COLOR 1,3: REM A BITTERKEPES
   KEPERNYO ELOTERSZINE = VOROS
30 COLOR 4,1: REM A KERETSZIN
   = FEKETE
```

Ebben a példában a háttér fekete, az előtér vörös és a keret fekete.

Váltás grafikus módba

A következő programlépés, hogy kiválasszuk a megfelelő grafikus módot. Ez a GRAPHIC paranccsal történik, amelynek formátuma a következő:

GRAPHIC mód [,c],s] vagy GRAPHIC CLR

ahol a *mód* 0 és 5 közötti szám, a *c* vagy 0, vagy 1, az *s* pedig egy 0 és 25 közötti érték. A 6.4. táblázatból leolvashatja a grafikus módoknak megfelelő értékeket.

6.4. táblázat

Grafikus módok

Mód	Leírás
0	40 oszlopos standard szöveg
1	Standard bittérkép
2	Standard bittérkép (osztott képernyő)
3	Színes bittérkép
4	Színes bittérkép (osztott képernyő)
5	80 oszlopos szöveg

A CLR paraméter a CLEAR-t jelenti. A 6.5. táblázatban láthatók a CLEAR értékei.

6.5. táblázat
CLEAR paraméterek

c érték	Leírás
0	Ne töröld a grafikus képernyőt
1	Töröld a grafikus képernyőt

Amikor először futtatja a programját, bizonyára törölni akarja majd a grafikus képernyőt, tehát adjon *c*-nek 1 értéket a GRAPHIC parancsban. Amikor azonban már másodszor futtatja, előfordulhat, hogy fent akarja hagyni a képet a képernyőn, hogy ne kelljen újra megrajzolnia, ebben az esetben legyen a *c*=0.

Az *s* paraméter azt jelzi, hogy hol legyen a szövegképernyő kezdete (a kiválasztott sorszám utáni sorban). Ha elhagyja az *s* paramétert, és osztott képernyős módba vált (2 vagy 4), a szövegrész a 20–25-ös sorban lesz, a képernyő többi része bittérképes. Az *s* paraméterrel lehet változtatni a szöveg kezdősorát 1-től 25-ig. Ha az *s* értéke 0, a képernyő nem osztott, hanem a teljes képernyőn szöveg lesz.
A GRAPHIC parancs utolsó paramétere a CLR. Amikor először ír be bittérképes grafikus parancsot, a Commodore 128-as 9 K területet allokál a bittérképes képernyő-információhoz. 8 K a bittérkép adatainak van fenntartva, a további 1 K pedig a színadatoknak (video mátrix). Mivel 9 K meglehetősen nagy tárblokk, előfordulhat, hogy később más célra akarja használni a program során. Ehhez szükséges a CLR, mert ez újrarendezi a Commodore 128-as tárát és visszaadja a bittérképes képernyő számára fenntartott 9 K-t. A CLR formátuma a következő:

```
GRAPHIC CLR
```

Ha ezt a formátumot használja, hagyja el a GRAPHIC parancs többi paraméterét.
Adja hozzá programjához a következő parancsot, amelynek eredményeképpen a Commodore 128-as standard bittérképes módba kerül, a parancs továbbá kijelöl egy 8 K-s bittérképes képernyőt grafikák készítésére (plusz 1 K-t a színadatoknak).

```
40 GRAPHIC 1,1
```

A parancsban levő második 1-es törli a bittérképes képernyőt. Amennyiben ezt nem akarja, cserélje ki a második 1-est 0-ra, de akár el is hagyhatja.
Megjegyzés: Ha bittérképes módban van, és nem tud visszatérni a szövegképernyőhöz, nyomja meg a RUN/STOP és a RESTORE billentyűket egyszerre, vagy előbb az ESC billentyűt, majd az X-et, ekkor visszajön a 80 oszlopos képernyő. Bár grafikát a VIC (40 oszlopos) chippel lehet csak megjeleníteni, írni írhat grafikus programot 80 oszlopos formátumban is. Ha Commodore 1902-es dual (kettős) monitorja van és futás közben meg akarja nézni a grafikus programot, a monitor csúszókapcsolóját át kell állítani 40 oszlopos kijelzésre.

Grafika megjelenítése a képernyőn

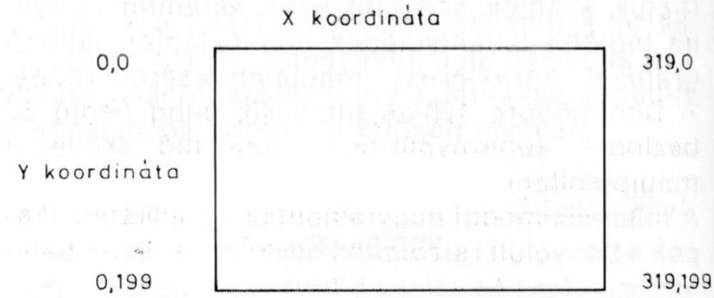
Eddig kiválasztotta a tetszés szerinti grafikus módot és színeket, most elkezdhet rajzolni a képernyőre. Kezdje egy körrel.

Kör rajzolása – a CIRCLE parancs

A kör rajzolásához a következő parancs szükséges:

```
60 CIRCLE 1,150, 130, 40,40
```

Erre a parancsra megjelenik a kör a képernyő közepén. A CIRCLE utasításnak kilenc paramétere van különböző körök és mértani alakzatok rajzolására. A 60-as sor CIRCLE utasításában a számok megváltoztatásával különböző méretű köröket és alakú körvonalakat (pl. oválist) lehet rajzolni. A CIRCLE utasítás sokoldalúbbá és hatékonyabbá teszi a grafikus BASIC programozást. A CIRCLE utasítás paramétereinek jelentését l. a BASIC 7.0 kislexikonban.
A Commodore 128-as képernyőjén az a pont, ahol X=0 és Y=0, a képernyő bal felső sarka, a HOME helyzet. Ugyanakkor a hagyományos geometriában az a pont, ahol X és Y is 0, a grafikon bal alsó sarka. A 6.1. ábráról leolvashatók az X (vízszintes) és Y (függőleges) képernyő-koordináták és a C128-as képernyő négy sarokpontja.



6.1. ábra. Az X és Y koordináták elhelyezkedése

Doboz rajzolása – a BOX parancs

Most rajzoljon egy dobozt! Ehhez írja be:

```
80 BOX1,20,100,80,160,90,1
```

Ezzel a paranccsal a kör bal oldalára egy doboz került. A BOX utasítás szá-

mainak jelentését a BASIC 7.0 kislexikon tartalmazza. Az utasításnak hét paramétere van, különböző típusú dobozok készítésére. Váltson előtérszínt és rajzoljon a kör jobb oldalára is egy dobozt, a következő utasításokkal:

```
90 COLOR1,9:REM ELOTERSZIN
VALTAS
100 BOX1,220,100,280,160,90,0
```

Kísérletezzen a BOX utasítással, rajzoljon különböző négyszögeket és dobozokat!

Vonalak, pontok és egyéb alakzatok rajzolása – a DRAW parancs

Már tudja, hogyan váltson grafikus módba, hogyan válassza meg a színeket, amelyekkel dolgozni kíván, és tud köröket, dobozokat rajzolni a képernyőre. A DRAW utasítással ugyanúgy lehet vonalat rajzolni a képernyőre, mint ceruzával papírra. A következő utasítás a dobozok és a kör alá egy vonalat húz:

```
120 DRAW 1,20,180 TO 280,180
```

A számok jelentése:

- 1 a színforrás (ebben az esetben az előtér)
- 20 a kezdő X (vízszintes) koordináta
- 180 a kezdő Y (függőleges) koordináta
- 280 az utolsó vízszintes koordináta
- 180 az utolsó függőleges képernyő-kordináta

Ha el akarja tüntetni a vonalat, cserélje fel az utasításban szereplő 1-est 0-ra. Ekkor a gép a háttér színével rajzolja meg a vonalat, amely így eltűnik. Gyakorolja különböző koordinátákkal a DRAW utasítást!

A DRAW utasításnak van egy másik formája is, amely lehetővé teszi, hogy rajzoljon egy vonalat, majd irányt változtatva rajzoljon egy másikat úgy, hogy a vonalak folyamatosak legyenek. Próbálja ki a következő utasítást:

```
130 DRAW 1,10,20 TO 300,20 TO 150,80 TO
10,20
```

Ez az utasítás egy háromszöget rajzolt a képernyő tetejére. A négy pár szám a háromszög három csúcsának X és Y koordinátáit jelöli. Bizonyára megfigyelte, hogy az első és utolsó koordináták azonosak, mivel a háromszöget ugyanazon a ponton kell befejezni, mint ahol elkezdtük rajzolni. A DRAW utasításnak ezzel a formájával szinte bármilyen mértani idomot – trapézt, paralelogrammát vagy sokszöget – lehet rajzolni.

Van a DRAW utasításnak egy harmadik formája is.

A kezdő X és Y értékek megadásával egyetlen pontot is lehet rajzolni:

```
150 DRAW 1,155,175
```

A képernyőn a kör alatt egy pont jelenik meg. Amint látja, a DRAW utasításnak olyan sokoldalú tulajdonságai vannak, hogy különböző pontok, vonalak és idomok szinte végtelen számú kombinációjának az előállítására képes.

Bekerített területek kifestése – a PAINT parancs

A DRAW utasítással különböző alakzatokat lehet elkeríteni a képernyőn. Mit tegyen, ha a vonalakkal határolt területeket ki akarja tölteni? Erre szolgál a PAINT (fest) utasítás, amellyel be lehet festeni az alakzatokat a 16 szín bármelyikével.

Írja be pl. a következőt:

```
160 PAINT 1,150,97
```

A 160-as sor befesti a 60-as sorban megrajzolt kört. A PAINT utasítás mindaddig kitölti a meghatározott területet, amíg a megjelölt forrás szerinti határt nem észleli. Amikor a Commodore 128-as befejezte a festést, azon a ponton hagyja a pixel kurzort, ahol a festést elkezdte. (Ez a fenti esetben a 150,97 pont.)

További két PAINT utasítás:

```
180 PAINT 1,50,25
200 PAINT 1,225,125
```

A 180-as sor a háromszöget, a 200-as sor pedig az üres dobozt festi ki.

Fontos tipp a festéshez: Ha a PAINT utasításban olyan kezdőpontot választ, amely már ugyanabból a forrásból be van festve, a Commodore 128-as nem festi be az illető területet. Olyan kezdőpontot kell választania, amely teljes egészében a befestendő terület határain belül helyezkedik el. A kezdőpont tehát nem lehet egy olyan pixel határvonalán, amelyet ugyanabból a forrásból már befestettünk. A képernyő-koordináta forrásszámai és a PAINT utasításban megjelölt koordináták nem lehetnek azonosak.

Karakterek megjelenítése a bittérképes képernyőn – a CHAR parancs

Mind ez idáig a mintaprogram standard bittérképes módban működött. A bittérképes mód a tárnak teljesen más területét használja a képernyő adatainak tárolására, mint a szöveg mód (az,

amelyben programokat és szövegeket írunk). Ha bittérképes módban próbál meg karaktereket kiírni, azt tapasztalja, hogy nem sikerül. Ez azért van így, mert a beírt karakterek a szövegképernyőn jelennek meg, Ön pedig a bittérképes képernyőt nézi. Előfordul, hogy táblázatok és grafikonok készítésénél szükség van arra, hogy a bittérképes képernyőre írjunk karaktereket. Ezt a célt szolgálja a CHAR parancs. Normál karaktereknek bittérképes képernyőn való megjelenítéséhez a következő CHAR utasítást használja:

```
220 CHAR 1, 11,24,"GRAFIKUS PELDA"
```

Ez az utasítás a "GRAFIKUS PELDA" szöveget a 12. oszlop 25. sorától kezdve írja ki. A CHAR parancsot lehet szöveg módban is használni, azonban elsősorban a bittérképes képernyőn használatos.

A grafikus alakzatok (formák) méretének megváltoztatása – a SCALE parancs

A Commodore 128-as grafikai rendszerének kibővítésére szolgál a SCALE utasítás, amellyel a képernyőn megjelenő alakzatokat lehet felnagyítani vagy kicsinyíteni. Van a SCALE utasításnak egy másik funkciója is, amelyet a következőkben ismertetünk:
Standard bittérképes módban a 40 oszlopos képernyőnek 320 vízszintes és 200 függőleges koordinátája van. Színes bittérképes módban a 40 oszlopos képernyő a standard bittérképes módnak csak a fele vízszintes lehetőségével dolgozik (160*200). Ezt a felbontásbeli hátrányt kárpótolja az, hogy egy 8*8-as karakteres mátrixon belül 3 szín (plusz egy szín) lehetőség jut, ui. standard bittérképes módban egy 8*8-as karakteres mátrixon belül csak két színt lehet megjeleníteni.
Ha a SCALE utasítást használja, a standard és a színes bittérképes módnak egymással arányosak a koordinátái, a vízszintes koordináták tartománya 1-től 1023-ig terjed. Ez a standard bittérképes és a színes módra egyaránt érvényes.
A SCALE utasítás használatához írja be a következőt:

```
SCALE 1,x,y
```

A képernyő-koordináták tartománya 0-tól 65 535-ig terjed standard vagy színes, nagyfelbontású módban egyaránt. A SCALE kikapcsolásához szükséges parancs a következő:

```
SCALE 0
```

és a koordináták visszatérnek normál értékeikhez.

Grafikus mintaprogram készítése

Mostanára már megismerkedett néhány grafikus utasítással. Rakja össze a program sorait és nézze meg, hogyan működnek egyszerre. A következőkben ismertetjük a program jelenlegi formáját. A 70-es, 110-es, 140-es, 170-es, 190-es és 210-es sorban a színutasítások minden tárgynak más színt adnak.

```
10 REM **** P.3 ****
20 COLOR 0,1 :REM PAPIR SZIN
30 COLOR 1,3 :REM TINTA SZIN
40 COLOR 4,1 :REM KERET SZIN
50 GRAPHIC 1,1: REM FINOM GRAFIKA VALASZTASA
60 CIRCLE ,150,130,40,40: REM KOR
70 COLOR 1,6: REM TINTA VALTASA
80 BOX ,20,100,80,160,90,1: REM DOBOZ
90 COLOR 1,9: REM TINTA VALTASA
100 BOX ,220,100,280,160,90,0: REM DOBOZ
110 COLOR 1,8: REM TINTA VALTASA
120 DRAW 1,20,180 TO 280,180: REM EGYENES VONAL
130 DRAW 1,10,20 TO 300,20 TO 150,80 TO 10,20: REM HAROMSZOG
140 COLOR 1,15: REM TINTA VALTASA
150 DRAW 1,150,175: REM  1 PONT
160 PAINT 1,150,97: REM KOR FESTESE
170 COLOR 1,5: REM TINTA VALTASA
180 PAINT 1,50,25: REM HAROMSZOG FESTESE
190 COLOR 1,7: REM TINTA VALTASA
200 PAINT 1,225,125: REM DOBOZ FESTESE
210 COLOR 1,11: REM TINTA VALTASA
220 CHAR ,11,24,"GRAFIKUS PELDA": REM KEPERNYO SZOVEG
230 FOR I=1 TO 5000:NEXT:GRAPHIC0,1:COLOR1,2
```

Most nézzük lépésről lépésre, mit csinál a program:

- a 20–40-es sorban kiválasztja a háttér, az előtér (szemléletesen: a papír és a tinta) és a keret színeit;
- az 50-es sor grafikus módba vált;
- a 60-as sor kört rajzol;
- a 80-as sor kifestett dobozt (vagy blokkot) rajzol;
- a 100-as sor megrajzolja egy doboz körvonalait;
- a 120-as sor egyenes vonalat rajzol a képernyő alján;
- a 130-as sor háromszöget rajzol;
- a 150-es sor egyetlen pontot rajzol a kör alá;
- a 160-as sor kifesti a kört;
- a 180-as sor kifesti a háromszöget;
- a 200-as sor kifesti az üres dobozt;
- a 220-as sor a "GRAFIKUS PELDA" karaktereket írja ki a képernyő aljára;
- a 230-as sor késlelteti a programot, hogy Ön megnézhesse a művét, visszakapcsol szöveg módba és feketére színezi a karaktereket.

Ha azt akarja, hogy a grafika a képernyőn maradjon, hagyja el a 230-as sorban a GRAPHIC utasítást.

Most néhány mintaprogram következik az imént tanult grafikus utasítások gyakorlására:

```
10 REM **** P.4 ****
20 COLOR 0,1
30 COLOR 1,8
40 COLOR 4,1
50 GRAPHIC1,1
60 FOR I=80 TO 240 STEP 10
70 CIRCLE 1,I,100,75,75
80 NEXT
90 COLOR 1,5
100 FOR I=80 TO 250 STEP 10
110 CIRCLE 1,I,100,50,50
120 NEXT
130 COLOR 1,7
140 FOR I=50 TO 280 STEP 10
150 CIRCLE 1,I,100,25,25
160 NEXT
170 FOR I=1 TO 7500:NEXT:GRAPHIC0,
:COLOR1,2
```

```
10 REM **** P.5 ****
20 GRAPHIC1,1
30 COLOR0,1
40 COLOR4,1
50 FOR I=1 TO 50
60 Z=INT(((RND(1))*16)+1)*1
70 COLOR 1,Z
80 X=INT(((RND(1))*30)+1)*10
90 Y=INT(((RND(1))*20)+1)*10
100 U=INT(((RND(1))*30)+1)*10
110 V=INT(((RND(1))*20)+1)*10
120 DRAW ,X,Y TO U,V
130 NEXT
140 SCNCLR
150 GOTO 50
```

```
10 REM **** P.6 ****
20 COLOR 4,7:COLOR 0,7:COLOR1,1
30 GRAPHIC1,1
40 FOR I=400 TO 1 STEP -5
50 DRAW 1,150,100 TO I,1
60 NEXT
70 FOR I=1 TO 400 STEP 5
80 DRAW 1,150,100 TO 1,I
90 NEXT
100 FOR I=40 TO 320 STEP 5
110 DRAW 1,150,100 TO I,320
120 NEXT
130 FOR I=320 TO 30 STEP -5
140 DRAW 1,150,100 TO 320,I
150 NEXT
160 FOR I=1 TO 7500:NEXT:GRAPHIC0,
1:COLOR 1,1
```

Írja be a programokat a számítógépébe, és futtassa, majd mentse ki, hogy bármikor újra használhassa őket. A programozás elsajátításának egyik legjobb módja, ha tanulmányozza a mintaprogramokat, és azt, hogy miként hajtják végre az utasítások a feladataikat. Nemsokára már Ön is fog tudni grafikus utasítások segítségével tetszetős grafikákat létrehozni a Commodore 128-ason.

További felvilágosításért lapozza fel a BASIC 7.0 kislexikont.

Most már ismer egy sor grafikus utasítást, amelyek használata lehetővé teszi, hogy szinte végtelen mennyiségben készítsen grafikákat. A Commodore 128-as azonban ennél többet is tud. Van még egy utasítássorozata, amelyet sprite-grafikának hívnak, és amellyel gyorsan, könnyedén és pontosan lehet a grafikus alakzatokat vezérelni. Ezekkel a magas szintű utasításokkal mozgatható grafikus tárgyakat, ún. sprite-okat lehet létrehozni, mert a számítógépek beépített SPRite DEFiniáló képessége van. Olvassa le a következő részt, és tegye meg az első lépéseket a számítógépes animáció elsajátítása felé!

A SPRITE-OK: MOZGATHATÓ, PROGRAMOZHATÓ TÁRGYAK

Megtanulta már, hogyan lehet a Commodore 128-assal különböző síkidomokat – köröket, dobozokat, vonalakat, pontokat – rajzolni; kifesteni a

képernyőt és az alakzatokat, átkapcsolni grafikus üzemmódba stb. Itt az ideje, hogy egy lépéssel előbbre lépjen a grafikus programozásban: a következőkben a sprite animációról lesz szó.
A fejezetben a következő utasításokkal fogunk foglalkozni:

MOVSPR
SPRDEF
SPRITE
SPRSAV
SSHAPE

Sprite-ok készítése

Először is el kell dönteni, hogyan nézzen ki a sprite, legyen mondjuk űrhajó vagy versenyautó. Mielőtt ki tudná színezni vagy mozgatni tudná a sprite-ot, először a formáját kell megterveznie. C128-as módban háromféleképpen lehet sprite-ot készíteni:

1. program módban, az új SPRITE utasításokkal;
2. SPRDEF módban;
3. ugyanúgy, mint a Commodore 64-essel.

Sprite-utasítások a programban

Ez a módszer beépített utasításokkal dolgozik, úgyhogy nincs szükség semmilyen programon kívüli segítségre, mint a másik két esetben. A módszer néhány grafikus utasítást az előzőkben tanultakból is alkalmaz. Előbb az általános leírás következik, a részleteket menet közben ismertetjük.

1. Rajzoljon egy képet az előző részben megismert grafikus utasítások segítségével (DRAW, CIRCLE, BOX és PAINT). Standard bittérképes módban a kép szélessége 24 pont (pixel), magassága 21 pont (pixel), színes bittérképes módban pedig szélessége 12 pont, magassága 21 pont.
2. A SSHAPE utasítással tárolja a kép adatait füzérváltozóban.
3. Vigye át a kép adatait a füzérváltozóból egy sprite-ba a SPRSAV utasítással.
4. Kapcsolja be (aktivizálja) a sprite-ot, színezze ki, váltson standard vagy színes módba és nagyítsa ki a sprite-ot a SPRITE utasítással.
5. Mozgassa a sprite-ot a MOVSPR utasítással.

A sprite képének megrajzolása

Most a sprite-működéshez szükséges utasítások következnek. Mire ezt a részt befejezi, már kész is az első sprite-programja. Akárhányszor futtathatja a programot vagy kimentheti későbbi felhasználásra.
Az első lépés, hogy egy képet rajzoljon (24*21 pont) a képernyőre a DRAW, a CIRCLE, a BOX és a PAINT utasításokkal. A mintaprogram standard bittérképes módban van, fekete háttérrel. A következő utasítás grafikus módba vált, és feketére festi a hátteret.

```
20 COLOR 0,1:REM HATTER FEKETE
30 GRAPHIC 1,1 :REM STANDARD
BITTERKEPES MOD
```

A most következő utasításokkal egy versenyautó képét rajzoljuk fel a bal felső sarokba. Az utasításokat már az előző rész során megtanultuk.

Futtassa a programot. Az imént egy fehér versenyautót rajzolt dobozba zárva a képernyő bal felső sarkába. A képernyő aljára versenypályát rajzolt, célvonallal. Ebben a pillanatban a versenyautó még csak álló kép, még nem sprite, azonban ez a forma megteremtése a sprite-programozás első lépése.

```
10 REM **** P.7 ****
20 COLOR 0,1:COLOR 4,1:COLOR 1,2
30 GRAPHIC1,1
40 BOX 1,2,2,45,45
50 DRAW 1,17,10 TO 28,10 TO 26,30 TO 19,30 TO 17,10: REM KOCSI TORZSE
60 DRAW 1,11,10 TO 15,10 TO 15,18 TO 11,18 TO 11,10: REM BAL FELSO KEREK
70 DRAW 1,30,10 TO 34,10 TO 34,18 TO 30,18 TO 30,10: REM JOBB FELSO KEREK
80 DRAW 1,11,20 TO 15,20 TO 15,28 TO 11,28 TO 11,20: REM BAL ALSO KEREK
90 DRAW 1,30,20 TO 34,20 TO 34,28 TO 30,28 TO 30,20: REM JOBB ALSO KEREK
100 DRAW 1,26,28 TO 19,28
110 BOX 1,20,14,26,18,90,1
120 BOX 1,150,35,195,40,90,1: REM UT
130 BOX 1,150,135,195,140,90,1: REM UT
140 BOX 1,150,215,195,220,90,1: REM UT
150 DRAW 1,50,180 TO 300,180:DRAW 1,50,180 TO 50,190:DRAW 1,300,180
TO 300,190
160 DRAW 1,50,190 TO 300,190
170 CHAR 1,20,23,"VEGE"
```

A sprite adatainak tárolása a SSHAPE utasítással

A következő lépés, hogy a képet átmentjük szövegfüzérbe a SSHAPE utasítással:

```
45 SSHAPE A$,10,11,34,31:REM KEP
MENTESE FUZERBE
```

A SSHAPE parancs füzérváltozóban tárolja a képet (bitminta) későbbi feldolgozásra, a kijelölt képernyő-koordináták szerint.
A 10, 11, 34 és 31 számok a kép koordinátái. A koordinátákat jól kell elhelyezni, különben az SSHAPE utasítással nem tudjuk helyesen tárolni a kép adatait az A$ füzérváltozóban. Ha az SSHAPE utasítást egy üres képernyőhelyre állítjuk be, az adatfüzér is üres lesz. Ha később sprite-tá változtatjuk, rájövünk, hogy nincsenek adatok. Győződjön meg arról, hogy az SSHAPE utasítást közvetlenül a helyes koordinátára állította be, valamint arról is, hogy a kép 24*21 pont legyen, ez egyetlen sprite mérete.
Az SSHAPE utasítás a versenyautó képét egy adatfüzérré változtatja, amelyet a számítógép képi adatként kezel. Az A$ adatfüzér a gép tárában egy nullákból és egyesekből álló füzért tárol, ezekből áll össze a képernyőn a kép. A számítógépes grafikában a vizuális grafikát a gép bitekkel reprezentálja a tárban. A képernyő minden pontjának (pixel) a gép tárában egy bit felel meg, amely vezérli. Standard bittérképes módban, ha a bit a tárban 1, akkor a pont a képernyőn bekapcsol, ha a tár vezérlőbitje 0, a pont kikapcsol.

Képi adatok átmentése a sprite-ba

A kép most egy füzérben van tárolva. A következő lépés az lesz, hogy az adatfüzér (A$) képi adatait átvisszük a sprite adatterületére, hogy be lehessen kapcsolni és mozgatni. Erre szolgál az SPRSAV utasítás:

```
50 SPRSAV A$,1:REM ADATFUZER TAROLAS
AZ 1. SPRITE-BA
55 SPRSAV A$,2:REM ADATFUZER TAROLAS
A 2. SPRITE-BA
```

A képi adatokat most átvittük az 1. és 2. sprite-ba. Mindkettőnek ugyanazok az adataik, tehát azonos képet mutatnak. Még nem láthatja a sprite-ot, előbb be kell kapcsolni (aktiválni).

A sprite bekapcsolása

A SPRITE utasítás bekapcsol egyet az 1-től 8-ig számozott sprite-ok közül, kiszínezi, meghatározza a képernyőprioritását, kinagyítja a méreteit és meghatározza a sprite-kijelzés típusát. A képernyőprioritás azt jelenti, hogy a sprite a képernyőn látható tárgyak előtt vagy mögött halad-e el. A sprite-okat eredeti nagyságuk kétszeresére lehet növelni mind vízszintesen, mind függőlegesen. A kijelzés típusa azt határozza meg, hogy a sprite standard vagy színes bittérképes-e. Következik a sprite-okat bekapcsoló két utasítás:

```
60 SPRITE 1,1,7,0,0,0,0;REM 1.SPRITE
BEKAPCS
65 SPRITE 2,1,3,0,0,0,0:REM 2.SPRITE
BEKAPCS
```

A számok jelentése:

SPRITE #,O,C,P,X,Y,M

#	a sprite száma (1–8);
O	be/ki kapcsolás (O = 1 be, O = 0 ki);
C	szín(1–16);
P	prioritás – ha P = 0, a sprite a többi tárgy előtt van, ha P = 1, a sprite a többi tárgy mögött van;
X	ha X = 1, a sprite vízszintes (X) irányban nő, ha X = 0, a sprite normál vízszintes méretű;
Y	ha Y = 1, a sprite függőleges irányban nő, ha Y = 0, a sprite normál függőleges méretű;
M	ha M = 1, a sprite színes, ha M = 0, a sprite standard.

Amint látja, a SPRITE utasítás igen széles hatókörű, és igen sok sprite tulajdonságot lehet vele vezérelni.

A sprite mozgatása a MOVSPR utasítással

Most, hogy a sprite már a képernyőn van, nincs más dolgunk, mint mozgatni. A MOVSPR utasítás vezérli a sprite mozgását a képernyőn. A MOVSPR utasítást kétféleképpen használjuk.

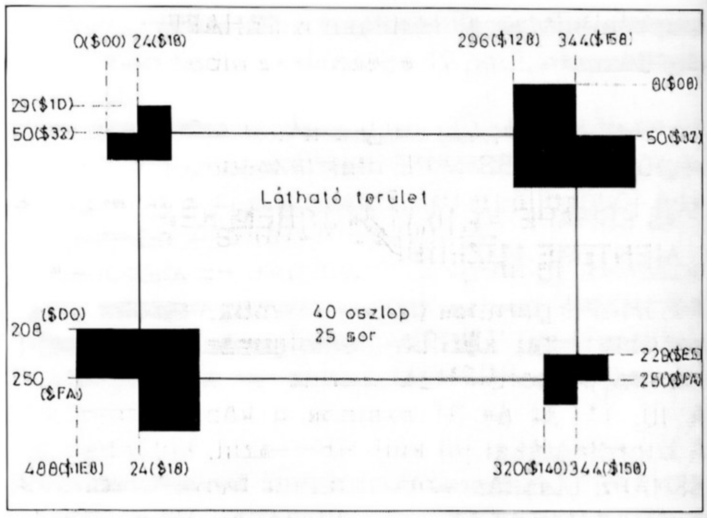
1. Függőleges és vízszintes koordinátákkal a képernyő abszolút helyére tehetjük a sprite-okat:
   ```
   70 MOVSPR 1,240,70:REM AZ 1.SPRITE HELYE
   X=240, Y=70
   80 MOVSPR 2,120,70:REM A 2.SPRITE HELYE
   X=120, Y=70
   ```
   A 70-es sor az 1. sprite-ot a (240,70) koordináták által meghatározott helyre teszi, a 80-as sor a 2. sprite-ot pedig (120,70)-re.

2. A MOVSPR utasítással a sprite-okat eredeti helyzetükhöz viszonyítva is lehet mozgatni. Helyezzük el pl. a sprite-okat a 70-es és 80-as sor szerint. Ha el akarjuk őket mozdítani a képernyő egy másik pontjára, használjuk pl. a következő utasításokat:

```
85 MOVSPR 1,180#6:REM AZ 1. SPRITE
FENTROL LEJON
90 MOVSPR 2,180#7:REM A 2. SPRITE
FENTROL LEJON
```

Az utasítás első száma a sprite száma. A második az irány, fokokban kifejezve az óramutató járásával megegyezően, a sprite eredeti helyzetéhez viszonyítva. A # jel azt mutatja, hogy a sprite kiindulási helyzetéhez képest meghatározott szögben és sebességgel mozog, nem pedig (mint a 70-es és 80-as sorban) abszolút helyen áll. A végső szám a sprite sebességét jelöli 0-tól 15-ig.

A MOVSPR parancsnak két formája van, l. a BASIC 7.0 kislexikont.
A sprite-ok a bittérkép koordinátáktól teljesen eltérő koordinátasíkban mozognak. A bittérkép koordinátái a (0,0) ponttól (bal felső sarok) a (319,199) pontig (jobb alsó sarok) terjednek. A látható sprite-koordináták az (50,24) pontnál kezdődnek és a (250,344) pontban végződnek. A többi sprite-koordináta nincs a képernyőn, tehát nem látható, de a sprite mégis aszerint mozog. A nem a képernyőn levő helyek segítségével tudnak a sprite-ok simán lejönni a képernyőről, majd visszamenni. A 6.2. ábra a sprite-koordinátasíkot és a látható sprite-helyzeteket mutatja.



6.2. ábra. A látható sprite-koordináták

Most futtassa az egész programot az összes lépéssel. Épp az imént írta meg első sprite-programját. Rajzolt egy versenypályát két autóval. Próbáljon meg további autókat, ill. tárgyakat felrajzolni a képre! Kísérletezzen más sprite-okkal is a versenypályán! Már a legjobb úton halad a sprite-programozás felé. Használja a fantáziáját, és találjon ki más jeleneteket és tárgyakat is, mozgassa őket, míg végül egy egész számítógépes „filmet" állít össze.

```
1 REM **** P.8 ****
5 COLOR 0,1:COLOR 4,1:COLOR 1,2
10 GRAPHIC1,1
15 BOX 1,2,2,45,45
20 DRAW 1,17,10 TO 28,10 TO 26,30 TO 19,30 TO 17,10: REM KOCSI TORZSE
22 DRAW 1,11,10 TO 15,10 TO 15,18 TO 11,18 TO 11,10: REM BAL FELSO KEREK
24 DRAW 1,30,10 TO 34,10 TO 34,18 TO 30,18 TO 30,10: REM JOBB FELSO KEREK
26 DRAW 1,11,20 TO 15,20 TO 15,28 TO 11,28 TO 11,20: REM BAL ALSO KEREK
28 DRAW 1,30,20 TO 34,20 TO 34,28 TO 30,28 TO 30,20: REM JOBB ALSO KEREK
30 DRAW 1,26,28 TO 19,28
32 BOX 1,20,14,26,18,90,1
35 BOX 1,150,35,135,40,90,1: REM UT
37 BOX 1,150,135,135,140,90,1: REM UT
40 BOX 1,150,215,135,220,90,1: REM UT
42 DRAW 1,50,180 TO 300,180:DRAW 1,50,180 TO 50,190:DRAW 1,300,180 TO 300,190
43 DRAW 1,50,190 TO 300,190
44 CHAR 1,20,23,"VEGE"
45 SSHAPE A$,11,10,34,31: REM SPRITE TAROLAS A$-BAN
50 SPRSAV A$,1: REM 0. SPRITE ADATA
55 SPRSAV A$,2: REM 1. SPRITE ADATA
60 SPRITE 1,1,7,0,0,0,0
65 SPRITE 2,1,3,0,0,0,0
70 MOVSPR 1,240,70
80 MOVSPR 2,120,70
85 MOVSPR 1,180 # 6
90 MOVSPR 2,180 # 7
95 FOR I=1 TO 5000:NEXT
99 GRAPHIC0,1
```

Ha a sprite-okat meg akarja állítani, nyomja meg egyszerre a RUN/STOP és a RESTORE billentyűt.

Sprite-program készítése

Most már van egy működőképes sprite-programja. Nézze meg a komplett programlistát:

A program menete lépésről lépésre:

- az 5-ös sor feketére festi a képernyőt;
- a 10-es sor beállítja a standard grafikus módot;
- a 15-ös sor egy dobozt rajzol a képernyő bal felső sarkába;
- a 20–32-es sor megrajzolja a versenyautót;
- a 35–44-es sor megrajzolja a pályát és a célvonalat;
- a 45-ös sor átviszi a versenyautó képadatait egy füzérváltozóba;
- az 50-es és az 55-ös sor a füzérváltozó tartalmát átviszi az 1. és 2. sprite-ba;
- a 60-as és 65-ös sor bekapcsolja a sprite-okat;
- a 70-es és a 80-as sor elhelyezi a sprite-okat a képernyő tetején;
- a 85-ös és a 90-es sor animálja a sprite-okat, mintha két autó száguldana át a célvonalon.

Ebből a részből megtanulta, hogyan alkothat sprite-okat a C128-as beépített grafikus utasításaival (DRAW és BOX). Tudja már vezérelni a sprite-okat a Commodore 128-as sprite-utasításaival. A Commodore 128-as további kétféle módon képes sprite-okat készíteni: vagy a beépített SPRite DEFinition (sprite-meghatározás) utasítással vagy a Commodore 64-es számítógépen alkalmazott eljárással azonos módon (l. a Commodore 64-es programozói kézikönyvét).

A sprite definíciós mód – a SPRDEF parancs

A Commodore 128-asnak beépített sprite-készítő módja van, a SPRDEF mód. Talán ismeri a Commodore 64-es sprite-készítési eljárását, amelynél vagy egy további sprite-szerkesztőre volt szükség, vagy meg kellett terveznie a sprite-ot egy rácsos papíron és beolvasni a kódolt adatokat, majd a POKE utasítással bevinni a sprite-blokkba. A Commodore 128-assal egyszerűbb a dolog, az új SPRDEF paranccsal egy speciális munkaterületen lehet tervezni és szerkeszteni sprite-okat. A SPRDEF módba úgy kell váltani, hogy beírja a SPRDEF parancsot, majd megnyomja a RETURN billentyűt. A képernyőn megjelenik a sprite-rács, valamint a kérdés:

SPRITE NUMBER? (sprite szám?)

Írjon be egy 1 és 8 közötti számot. A számítógép kitölti a rácsot és kiírja a megfelelő sprite-ot a képernyő jobb felső sarkába. A továbbiakban a sprite-rácsot munkaterületnek fogjuk nevezni. Ez 24 karakter széles és 21 karakter magas. A munkaterület minden egyes karakterhelye a sprite egy pontjának felel meg, mivel a sprite 24 pont széles és 21 pont magas. Míg SPRDEF módban dolgozik a munkaterületen, néhány szerkesztési parancs áll rendelkezésére. A parancsok a következők:

CLR billentyű	– törli az egész munkaterületet;
M billentyű	– be/ki kapcsolja a színes sprite-ot;
CTRL 1–8	– meghatározza a sprite előtérszínét (1–8);
C= 1–8	– meghatározza a sprite előtérszínét (9–16);
1 billentyű	– bekapcsolja a háttérszín pontjait;
2 billentyű	– bekapcsolja az előtérszín pontjait;
3 billentyű	– bekapcsolja a színes 1 területeket;
4 billentyű	– bekapcsolja a színes 2 területeket;
A billentyű	– automata kurzor ki/be kapcsoló;
CRSR billentyűk	– a kurzort mozgatják a munkaterületen;
RETURN	– a kurzort a következő sor elejére viszi;
HOME billentyű	– a kurzort a munkaterület bal felső sarkába viszi;
X billentyű	– vízszintesen nagyítja a sprite-ot;
Y billentyű	– függőlegesen nagyítja a sprite-ot;
SHIFT RETURN	– kimenti a sprite-ot a munkaterületről és visszatér a SPRITE NUMBER promthoz;
C billentyű	– az egyik sprite-ot rámásolja a másikra;
STOP billentyű	– kikapcsolja a kiírt sprite-ot és annak megváltoztatása nélkül visszatér a SPRITE NUMBER üzenethez;
RETURN billentyű	– (a SPRITE NUMBER üzenet után) kilép a SPRDEF módból.

Sprite készítése SPRDEF módban

A sprite-készítés általános menete:

1. Töröljük a képernyőt a SHIFT és a CLR/HOME billentyűk együttes lenyomásával.

2. Ha színes sprite-ot akar, nyomja meg az M billentyűt és az eredeti kurzor mellett megjelenik még egy. Azért jelenik meg két kurzor, mert a színes módban két pont van bekapcsolva a standard sprite mód egy pontjával szemben. Ezért van színes módban csak a fele vízszintes lehetőség, mint a standard módban.

3. Válassza ki a sprite színét. Az 1–8 színekhez tartsa lenyomva a CONTROL billentyűt és nyomjon le egy billentyűt 1-től 8-ig. A 9–16 színkódokhoz tartsa lenyomva a C= billentyűt és így nyomjon meg egyet 1-től 8-ig.

4. Most hozzá lehet kezdeni a sprite megtervezéséhez. Az 1–4 billentyűk kitöltik a sprite-ot és megadják az alakját. Egyszínű sprite-hoz a 2-es billentyűvel töltse ki a karakterhelyet a munkaterületen. Az 1-es billentyűvel lehet kitörölni azt, amit a 2-essel rajzolt. Ha egyszerre egy karakterhelyet akar kitölteni, nyomja meg az A billentyűt. Most a kurzort kézzel, a kurzorbillentyűkkel kell mozgatnia. Ha azt akarja, hogy a kurzor automatikusan haladjon jobbra, míg lenyomva tartja a billentyűt, ne nyomja le az A billentyűt, mert az már automata kurzormozgatásra van beállítva. Ahogy kitölt egy karakterhelyet a munkaterületen, látja, amint a megfelelő pont megjelenik a kijelzett sprite-on. A sprite-szerkesztés közvetlenül a munkaterületen való szerkesztés után történik. Színes módban a 3-as billentyű a munkaterület két karakterhelyét tölti ki a színes 1 színnel, míg a 4-es billentyű két helyet tölt ki a színes 2 színnel.

   A kitöltött területeket a háttérszínre az 1-es billentyűvel festheti át. Színes módban az 1-es billentyű egyszerre két karakterhelyet fest át.

5. Munka közben szabadon mozoghat a munkaterületen a RETURN, a HOME és a kurzorbillentyűk segítségével anélkül, hogy a képernyőpontokat ki- vagy bekapcsolná.

6. Sprite-jait bármikor kinagyíthatja vízszintes és függőleges irányban egyaránt. A függőleges irányú nagyításhoz nyomja meg az Y billentyűt, a vízszinteshez pedig az X-et. Ha vissza akar térni a normál mérethez, nyomja meg ismét az X vagy Y billentyűt. Ha ugyanaz a billentyű vezérli a ki- és bekapcsolást, ezt váltókapcsolónak (toggle) nevezzük, tehát az X és Y billentyűk kapcsolják a sprite vízszintes, ill. függőleges nagyítását-kicsinyítését.

7. Ha készen van a sprite-tal és tetszik, mentse ki a SHIFT és a RETURN billentyű egyidejű lenyomásával. A Commodore 128-as egy külön sprite-tárterületen tárolja a sprite-adatokat. A képernyő jobb felső sarkából eltűnik a sprite és a vezérlés visszatér a SPRITE NUMBER üzenethez. Ha újabb sprite-ot akar készíteni, írjon be egy új számot, és járjon el mindenben az első készítéséhez hasonlóan. Ha az eredetit látni szeretné a munkaterületen, írja be az eredeti sprite számát. Ha ki akar lépni az SPRDEF módból, a SPRITE NUMBER megjelenése után nyomja meg a RETURN-t.

8. A C billentyűvel lehet az egyik sprite-ot a másikra másolni.

9. Ha a sprite-okat nem akarja kimenteni, nyomja meg a STOP billentyűt. Ilyenkor a sprite eltűnik a képernyőről és megjelenik a SPRITE NUMBER üzenet.

10. Ha ki akar lépni az SPRDEF módból, nyomja meg a RETURN billentyűt, amíg a SPRITE NUMBER üzenet látható és nem írt be új számot. Kilépni a következő feltételek egyike mellett lehet:
    közvetlenül a sprite kimentése után (SHIFT-elt RETURN);
    közvetlenül a STOP billentyű lenyomása után.
    Ha készen van a sprite szerkesztésével és kilépett az SPRDEF módból, a sprite adatait a Commodore 128-as az arra kijelölt helyen tárolja. Mivel most visszakerült BASIC nyelvbe, be kell kapcsolnia (aktiválnia kell) a sprite-ot, hogy megjelenjen a képernyőn. Ehhez a SPRITE parancsot kell használni. Mondjuk, hogy SPRDEF módban csinált egy 1-es számú sprite-ot. Ha BASIC-ben be akarja kapcsolni, kékre festeni és mind X, mind pedig Y irányban kinagyítani, írja be a következő parancsot:

```
SPRITE 1,1,7,0,1,1,0
```

Most a MOVSPR paranccsal mozgassa 90 fokos szögben, 5-ös sebességgel:

```
MOVSPR 1, 90 # 5
```

Most már mindent tud a SPRDEF módról. Tehát: először is alkossa meg a sprite-ot, mentse ki az adatait, majd váltson át SPRDEF módból BASIC-be. Ezután kapcsolja be a sprite-ot a SPRITE paranccsal, és mozgassa a MOVSPR paranccsal. Ha kész a program, mentse a sprite-adatokat egy bináris file-ba a következő BSAVE parancs segítségével:

BSAVE "filenév", B0, P3584 TO 4096

Ha az adatokat a lemezről ismét le akarja hívni, töltse be az előzőleg kimentett bináris file-t a BLOAD paranccsal:

BLOAD "filenév"[, B0, P3584 TO P4096]

A zárójeles rész opcionális, a BLOAD abba a címbe tölti be az adatokat, amelyről kimentették. Kísérletezzen az új módszerekkel: 1) SSHAPE, SPRSAV, SPRITE, MOVSPR; 2) SPRDEF mód. Sajátítsa el a sprite-készítés technikáját.

További felvilágosítást az alfejezet végén levő „Sprite-adatok tárolása file"-okban c. részben talál.

Egymáshoz kapcsolódó sprite-ok

Már megtanulta, hogyan kell egy sprite-ot megszerkeszteni, kiszínezni, bekapcsolni és mozgatni. Előfordulhat azonban, hogy olyan képet akar készíteni, amely túl nagy ahhoz, hogy beleférjen egyetlen sprite-ba. Ebben az esetben össze lehet kapcsolni két vagy több sprite-ot, és a kép nagyobb és részletesebb lesz, mintha csak egy sprite-ot használna. Az összekapcsolt sprite-ok egymástól függetlenül is tudnak mozogni. Így sokkal jobban lehet vezérelni az animációt, mint egy sprite-tal.
A következőkben ismertetjük a két vagy több öszszekapcsolt sprite-tal készült program algoritmusát:

1. Rajzoljon egy képet a Commodore 128-as grafikus utasításaival (DRAW, BOX és PAINT) úgy, mint a versenypálya programban. Ezúttal azonban a kép kétszer akkora legyen, mint egy sprite (48*21 pont).
2. Két SSHAPE utasítással tárolja a sprite-okat két külön adatfüzérben. Helyezze az első SSHAPE koordinátákat a kép első (24*21 pont) felére, a másodikat pedig a második ugyanekkora területre. Nagyon fontos, hogy a képi adatokat külön füzérben tárolja. Pl. az első SSHAPE utasítás a kép első felét az A$-ban tárolja, a második SSHAPE utasítás a kép másik felét a B$-ban.
3. Vigye át mindkét adatfüzérről a képi adatokat egy-egy külön sprite-ba a SPRSAV utasítással.
4. Kapcsolja be a két sprite-ot a SPRITE utasítással.
5. Helyezze el úgy a sprite-okat, hogy az egyik ott kezdődjön, ahol a másik végződik, tehát a következő pontban. Ezzel a lépéssel kapcsolta össze a két sprite-ot. Pl. rajzoljon egy képet, amelynek méretei 48*12 pont. Az első sprite-ot (1) a (10,10) helyre tegye a következő utasítással:

   100 MOVSPR 1, 10,10

   ahol az első szám a sprite száma, a második szám a vízszintes koordináta (X), a harmadik szám a függőleges koordináta (Y). Vigyük most a második sprite 24 képpontját a

   200 MOVSPR 2,34 10

   utasítással az 1-es sprite jobb oldalához. Ebben a pillanatban a két sprite közvetlenül egymás mellett jelenik meg a képernyőn, és pontosan úgy néznek ki, ahogyan azokat a program elején a DRAW, BOX és PAINT utasításokkal megrajzoltuk.
6. Most a sprite-okat további MOVSPR utasításokkal tetszés szerint akár együtt, azonos irányba, akár külön-külön, különböző irányokba is lehet mozgatni. Amint azt már láttuk, a MOVSPR utasítással a sprite-okat áthelyezhetjük a képernyő általunk meghatározandó részére, vagy valamely, a sprite eredeti helyéhez viszonyított részére.

A következő program sprite-ok egymáshoz illesztésére mutat be egy példát. A program űrbéli környezetet készít. Felrajzol csillagokat, egy bolygót és egy, az Apollóhoz hasonló űrhajót. Miután megrajzolta az űrhajót, az erre vonatkozó adatokat elhelyezi az A$ és a B$ karakterfüzérekben. Az űrhajó elülső részét, az űrkabint az 1-es sprite, a hátulsó részét, a visszatéréshez szükséges rakétát a 2-es sprite tartalmazza. Az űrhajó kétszer, lassan átrepül a képernyőn. Mivel már ilyen lassan repül, és a Föld még nagyon messze van, meg kell kezdeni a visszatérést. Miután másodszor is áthaladt a képernyőn, begyújtja a visszatérést indító rakétáit, amelyek az űrkabint biztonságosan röpítik a Föld felé.

A program listája:

```
1 REM **** P.9 ****
5 COLOR 4,1:COLOR 0,1:COLOR 1,2: REM FEKETE KERET ES PAPIR, FEHER TINTA
10 GRAPHIC1,1: REM NAGYFELBONTASU KEPERNYO BEALLITASA
17 FOR I=1 TO 40
18 X=INT(RND(1)*320)+1: REM RAJZOLAS KEZDETE
19 Y=INT(RND(1)*200)+1: REM RAJZOLAS KEZDETE
21 DRAW 1,X,Y:NEXT: REM RAJZOLAS KEZDETE
22 BOX 0,0,5,70,40,,1: REM DOBOZ TORLESE
23 BOX 1,1,5,70,40: REM DOBOZ AZ URHAJOHOZ
24 COLOR 1,8:CIRCLE 1,190,90,35,25: PAINT 1,190,95: REM BOLYGO RAJZOLASA
ES FESTESE
25 CIRCLE 1,190,90,65,10: CIRCLE 1,190,93,65,10: CIRCLE 1,190,95,65,10:
COLOR 0,1
26 DRAW 1,10,17 TO 16,17 TO 32,10 TO 33,20 TO 32,30 TO 16,23 TO 10,23 TO 10,17
28 DRAW 1,19,24 TO 20,21 TO 27,25 TO 26,28: REM ALSO ABLAK
35 DRAW 1,20,19 TO 20,17 TO 29,13 TO 30,18 TO 28,23 TO 20,19: REM FELSO ABLAK
38 PAINT 1,13,20: REM URHAJO FESTESE
40 DRAW 1,34,10 TO 36,20 TO 34,30 TO 45,30 TO 46,20 TO 45,10 TO 34,10
  : REM 1. SPRITE
42 DRAW 1,45,10 TO 51,12 TO 57,10 TO 57,17 TO 51,15 TO 46,17: REM 1. HAJTOMU
43 DRAW 1,46,22 TO 51,24 TO 57,22 TO 57,29 TO 51,27 TO 45,29: REM 2. HAJTOMU
44 PAINT 1,40,15: PAINT 1,47,12:PAINT 1,47,26: DRAW 0,45,30 TO 46,20 TO 45,10
45 DRAW 0,34,14 TO 44,14:DRAW 0,34,21 TO 44,21: DRAW 0,34,28 TO 44,28
47 SSHAPE A$,10,10,33,32: REM SPRITE AZ A$-BA
48 SSHAPE B$,34,10,57,32: REM SPRITE AZ B$-BA
50 SPRSAV A$,1: REM 1. SPRITE ADATA
55 SPRSAV B$,2: REM 2. SPRITE ADATA
60 SPRITE 1,1,3,0,0,0,0: REM 1. SPRITE BEALLITASA
65 SPRITE 2,1,7,0,0,0,0: REM 2. SPRITE BEALLITASA
82 MOVSPR 1,150,150: REM 1. SPRITE EREDETI HELYZETE
83 MOVSPR 2,172,150: REM 2. SPRITE EREDETI HELYZETE
85 MOVSPR 1,270 # 5: REM 1. SPRITE MOZGATASA A KEPERNYON ATLOSAN
87 MOVSPR 2,270 # 5: REM 2. SPRITE MOZGATASA A KEPERNYON ATLOSAN
90 FOR I=1 TO 5950:NEXT: REM KESLELTETES
92 MOVSPR 1,150,150: REM 1. SPRITE ELOKESZULET A RAKETA KILOVESERE
93 MOVSPR 2,174,150: REM 2. SPRITE ELOKESZULET A RAKETA KILOVESERE
95 MOVSPR 1,270 # 10: REM RAKETA SZETVALIK
96 MOVSPR 2,90 # 5: REM RAKETA SZETVALIK
97 FOR I=1 TO 1200:NEXT: REM KESLELTETES
98 SPRITE 2,0: REM HATRA MOZGATAS VEGE
99 FOR I=1 TO 20500:NEXT: REM KESLELTETES
100 GRAPHIC0,1: REM VISSZA A SZOVEG MODBA
```

A program működése:

5-ös sor:	a COLOR utasítással a keret és a háttér színét feketére, a képernyőt fehérre állítja be.
10-es sor:	bekapcsolja a standard nagyfelbontású üzemmódot és törli a képernyőt.
17–21-es sorok:	felrajzolják a képernyőre a csillagokat.
23-as sor:	a képernyő bal felső sarkában kijelöl egy területet az űrhajó megrajzolásához (BOX).
24-es sor:	megrajzolja és befesti a bolygót (DRAW és PAINT).
25-ös sor:	köröket rajzol a bolygó köré (DRAW és CIRCLE).
26-os sor:	megrajzolja az űrhajó űrkabin részét (DRAW).
28-as sor:	megrajzolja az űrkabin alsó ablakát (DRAW).
35-ös sor:	megrajzolja az űrkabin felső ablakát (DRAW).
38-as sor:	az űrkabint fehérre festi (PAINT).
40-es sor:	megrajzolja az űrhajó rakéta részét (DRAW).
42-es és 43-as sor:	megrajzolják a rakéta hajtóműveit (DRAW).
44-es sor:	befesti a rakéta hajtóműveit és

megrajzolja a rakéta hátsó vonalát a háttér színében (PAINT, DRAW).

45-ös sor: vonalakat rajzol az űrhajó rakéta részére a háttér színében (DRAW). (A programmal idáig még csak rajzoltunk a képernyőre. Semmilyen sprite-ra vonatkozó utasítást nem használtunk, így ez az űrhajó még nem sprite.)

47-es sor: az SSHAPE utasítás a koordinátákat az űrhajó első felére, az űrkabinra pozicionálja (24*21 képpont), és az itt levő adatokat elhelyezi az A$ karakterfüzérben.

48-as sor: az SSHAPE utasítás a koordinátákat az űrhajó második részére pozicionálja (24*21 képpont), és az itt levő adatokat elhelyezi a B$ karakterláncban.

50-es sor: az A$-ból az adatokat átviszi az 1-es sprite-ba.

55-ös sor: a B$-ból az adatokat átviszi a 2-es sprite-ba.

60-as sor: bekapcsolja az 1-es sprite-ot, és pirosra színezi.

65-ös sor: bekapcsolja a 2-es sprite-ot, és kékre színezi.

82-es sor: az 1-es sprite-ot a 150, 150 koordinátapontra helyezi.

83-as sor: a 2-es sprite-ot az 1-es sprite kezdőpontjához képest 24 képpontnyi hellyel jobbra helyezi. A 82-es és a 83-as sorokban levő utasításokkal összekapcsoltuk a két sprite-ot.

85-ös és 87-es sor: az összekapcsolt sprite-okat átviszi a képernyőn.

90-es sor: késlelteti a program futását. A késleltetésre azért van itt szükség, hogy a sprite-oknak legyen elegendő idejük ahhoz, hogy képernyőn kétszer áthaladjanak. Ha kihagynánk a késleltetést, akkor a sprite-oknak nem lenne idejük arra, hogy áthaladjanak a képernyőn.

92-es és 93-as sor: a sprite-okat a képernyő közepére helyezi és előkészíti az űrhajót az indítórakéták begyújtására.

95-ös sor: elindítja az 1-es sprite-ot, azaz az űrkabint. A sorban szereplő 10-es szám a sprite mozgásának a sebességét határozza meg. A szám 1 és 15 között lehet. Ha 1, akkor a sprite áll, ha 15, akkor villámgyorsan mozog.

96-os sor: a felhasznált rakétát hátrafelé mozgatja, és kitolja a képernyőről.

97-es sor: szintén programkésleltetés. Azért kell, hogy a 2-es sprite-nak, a rakétának legyen ideje a képernyő elhagyására.

98-as sor: kikapcsolja a 2-es sprite-ot, miután az elhagyta a képernyőt.

99-es sor: másik programkésleltetés. Azért kell, hogy az űrkabinnak legyen ideje arra, hogy végighaladjon a képernyőn.

100-as sor: ezzel a sorral visszatérünk a karakteres üzemmódba.

Az összekapcsolt sprite-okkal érdekesebb a munka, mintha csak eggyel dolgoznánk. Összekapcsolt sprite-ok programozásánál főként a következőkre kell figyelni: 1. Győződjünk meg arról, hogy az SSHAPE utasítás koordinátáit a képernyő megfelelő helyére pozicionáltuk, hogy a rajz megfelelő adatait helyesen tároljuk; 2. győződjünk meg arról, hogy a sprite-oknak a MOVSPR utasítással történő összekapcsolásakor a helyes koordinátákat adtuk meg. Ebben a példában a 2-es sprite-ot az 1-es sprite kezdőpontjához képest 24 képpontnyi hellyel jobbra pozicionáltuk. Ha már elsajátította, hogyan kell két sprite-ot összekapcsolni, kísérletezzen kettőnél többel is. Minél többet kapcsol össze, annál pontosabbak lesznek a részletek és élénkebb az animáció.

A Commodore 128-asnak van még két sprite-parancsa, a SPRCOLOR és a COLLISION, amelyekkel ebben a fejezetben nem foglalkozunk, de megtalálja őket a BASIC 7.0 kislexikonban

Sprite-adatok tárolása bináris file-okban

Figyelem! Az itt következő magyarázat némi ismeretet feltételez a gépi nyelvről, a tárhelyekről, a bináris file-okról és a tárgykód file-okról.

A Commodore 128-as számítógépnek van két új parancsa, a BLOAD és a BSAVE, amelyekkel könnyen és jól tárolhatók a sprite-adatok. A parancsok nevében a B a bináris szóra utal. A BLOAD és a BSAVE parancs bináris file-okat ment lemezre, ill. tölt lemezről. A bináris file vagy egy gépi nyelvű program egy része, vagy egy meghatározott címtartományon belüli adatgyűjtemény. Valószínűleg ismeri a beépített gépi nyelvű monitor SAVE parancsát. Ha ezt a parancsot használja, a lemezen tárolt file bináris file. Ezzel könnyebb dolgozni, mint a tárgykód file-lal, mert a bináris file-t minden további előkészület nélkül be lehet tölteni, míg a tárgykód file-t töltővel (loader) kell tölteni, mint pl. a Commodore 64 Assembler Development Systemnél, ahol a SYS paranccsal lehet a töltést végrehajtani. Ha bináris file-okat akar betölteni, a következő eljárások valamelyikét kell alkalmaznia.

```
LOAD "bináris file-név", 8,1
```

vagy

BLOAD "bináris file-név", B0,Pstart

ahol a start=3584, ha sprite adatfile-okat akar betölteni.
Az első megoldásnál az 1-est ne felejtse el, mert különben a számítógép úgy kezeli, mint egy BASIC programfile-t és a BASIC szöveg elejére tölt be. Az 1 azt közli a számítógéppel, hogy a bináris file-t ugyanarra a helyre töltse be, mint amelyről kimentette.
Bizonyára kíváncsi, mi köze van mindennek a sprite-okhoz. Az összefüggés a következő: a C128-asnak van egy erre a célra kijelölt tára, amely a 3584 ($0E00) decimális címtől a 4095 ($0FFF) címig terjed, itt tárolja a sprite-adatokat. A tárnak ez a része 512 byte-ot vesz igénybe. Amint tudja, egy sprite 24 pont széles és 21 pont magas. Minden pont egy bitet foglal el a tárban. Ha egy sprite bitje ki van kapcsolva (egyenlő 0-val), a képernyőn a neki megfelelő pont is ki lesz kapcsolva, azaz felveszi a háttér színét. Ha egy sprite-on belül egy pont be van kapcsolva (egyenlő 1-gyel), akkor a képernyőn a neki megfelelő pont is be lesz kapcsolva, azaz az előtér színét veszi föl. A nullák és egyesek kombinációja adja a képernyőn megjelenő képet.
Mivel egy sprite 24*21 pontnyi, és minden pont egy bit helyet foglal el a tárban, egy sprite 63 byte tárat használ föl. A 6.3. ábra ismerteti a sprite-adatok tárolási szükségleteit.

```
    12345678  12345678  12345678
 1  ........  ........  ........
 2  ........  ........  ........
 3  ........  ........  ........
 4  ........  ........  ........
 5  ........  ........  ........
 6  ........  ........  ........
 7  ........  ........  ........
 8  ........  ........  ........
 9  ........  ........  ........
10  ........  ........  ........
11  ........  ........  ........
12  ........  ........  ........
13  ........  ........  ........
14  ........  ........  ........
15  ........  ........  ........
16  ........  ........  ........
17  ........  ........  ........
18  ........  ........  ........
19  ........  ........  ........
20  ........  ........  ........
21  ........  ........  ........
```

Minden sor = 24 bit = 3 byte

6.3. ábra. Sprite-adatok társzükséglete

Egy sprite-nak 63 byte adatra van szüksége. Tulajdonképpen minden sprite-blokk 64 byte-ból áll, de egy byte felhasználatlanul marad. Mivel a C128-asnak 8 sprite-ja van, és mindegyik egy 64 byte-os blokkból áll, a számítógépnek 512 (8*64) byte-ra van szüksége, hogy mind a nyolc sprite adatait tárolni tudja.
A nyolc sprite-blokk tárolási helye a 3584-es ($0E00) tárhelynél kezdődik és a 4095-ösnél ($0FFF) végződik. A 6.4. ábra a tárcímtartományokat ismerteti, ahol az egyes sprite-ok tárolják az adataikat.

```
$0FFF (4095 Decimal)
      ]—Sprite 8
$0FC0
      ]—Sprite 7
$0F80
      ]—Sprite 6
$0F40
      ]—Sprite 5
$0F00
      ]—Sprite 4
$0EC0
      ]—Sprite 3
$0E80
      ]—Sprite 2
$0E40
      ]—Sprite 1
$0E00 (3584 Decimal)
```

6.4. ábra A sprite-ok tárcímtartományai

BSAVE Ha kilépett a SPRDEF üzemmódból, bináris sprite file-okban tárolhatja sprite-adatait. Ily módon bármilyen sprite-kollekciót könnyedén vissza tud tölteni a számítógépbe. A következő paranccsal lehet a sprite-adatokat kimenteni bináris file-ba:

BSAVE "filenév",B0 P3584 TO P4096

A bináris file-név az a név, amit Ön ad a file-nak. A B0 azt jelzi, hogy 0 bankból menti ki a sprite-adatokat. A P3584 TO P4096 paraméterek azt jelzik, hogy a 3584–4096 címtartományt menti ki, amely az összes sprite-adat tárolási helye.
Nem kell minden sprite-ot definiálni, amikor kimenti őket. A definiált sprite-okat a helyes sprite-blokkból menti ki a gép, a nem definiált sprite-okat szintén bináris file-ban a megfelelő sprite-blokkból, azonban a gép nem törődik velük, ui. könnyebb kimenteni mind a 8 sprite 512 byte-ját, függetlenül attól, hogy mindet használjuk-e, mint külön kimenteni az egyes blokkokat.

BLOAD Később, ha ismét fel akarja használni a sprite-okat, csak be kell tölteni (BLOAD)

az összes sprite mind az 512 byte-ját a 3584 ($0E00)–4096 ($0FFF) tartományba a következő paranccsal:

```
BLOAD "filenév"[,B0,B3584]
```

Ugyanezt a file-nevet kell használnia, mint amit az eredeti sprite-adatok kimentésénél használt. A B0 a 0 bankszámra utal, a P3584 pedig a bináris sprite-adatfile kezdő tárolási helyére. Az utolsó két paraméter opcionális.

****

A most véget ért alfejezetből kiderült, hogy mennyire le tudják egyszerűsíteni az új Commodore 7.0 BASIC parancsok a grafikus képek alkotásának és mozgatásának általában bonyolult folyamatát. A következő részben szintén új BASIC 7.0 parancsokkal ismerkedhet meg, amelyek ugyanezt teszik a zene és a hanghatások területén.

7. ALFEJEZET

Hang és zene C128-as üzemmódban

BEVEZETÉS

A SOUND utasítás
Írjunk hangprogramot 75
Véletlenhangok 76

Bonyolult hang- és zenei hatások C128-as üzemmódban
Rövid összefoglalás – a hang tulajdonságai 77
Zenélés a Commodore 128-assal 78
A zenei program összeállítása 82
Magas fokú szűrés 83
Dallam kódolása kottából

BEVEZETÉS

A Commodore 128-as számítógép beépített hangszintetizátora a mikroszámítógépek világában az egyik legfejlettebb. A szintetizátor, amelynek neve Sound Interface Device (SID), egy chip, amely kizárólag hang és zene generálására alkalmas. A SID chip egyszerre három független hang (szólam, oszcillátor) képzésére képes. Minden szólamot négy hangtípus (hullámforma) valamelyikében lehet lejátszani. A SID chipnek programozható ADSR paraméterei vannak. (Attack – Decay – Sustain – Release = felfutás – lecsengés – kitartás – elengedés.) Ezek a paraméterek határozzák meg a hang minőségét. Ezen kívül a szintetizátornak van szűrője is, amellyel bizonyos hangokat kiválaszthat, míg másokat kizár, vagy módosítja a hang vagy hangok jellegét.
A következő rész ezeket a paramétereket ismerteti, segítségükkel szinte bármilyen hangot elő lehet állítani. A SID chipek számos lehetőségének kihasználását könnyítik meg az új BASIC zenei utasítások:

SOUND
ENVELOPE
VOL
TEMPO
PLAY
FILTER

A következőkben egyenként tárgyaljuk ezeket az utasításokat és felépítünk belőlük egy zenei mintaprogramot. Mire az alfejezet végére ér, ismerni fogja a zenei programok készítésének módját. A mintaprogram segítségével saját, bonyolult szerzeményeket játszó programokat is írhat. Beprogramozhatja a kottákat és saját készítésű hanghatásokkal játszhatja le a nagy klasszikusok, pl. Beethoven műveit, vagy akár a Beatles valamelyik dalát. Számítógépes zenével alá is festheti grafikus programjait, így akár egy kis házi videóra is szert tehet.

A SOUND UTASÍTÁS

A SOUND utasítást elsősorban gyors és egyszerű hanghatások elérésére használjuk. A komplett zenei hangszerelések bonyolultabb módjairól és a többi hangutasításról a későbbiekben lesz szó.

A SOUND utasítás formátuma a következő:

SOUND VC,FREQ, DUR[,DIR[,MIN[,SV[, WF[,PW]]]]]

A paraméterek jelentése:

VC a hang (szólam, oszcillátor) kiválasztása (1, 2 vagy 3)
FREQ a hang frekvenciaszintje (0–65 535)
DUR a hang tartama (60-ad másodpercekben)
DIR a hang iránya
0 = a frekvencia nő
1 = a frekvencia csökken
2 = a frekvencia oszcillál (fel-le)
MIN beállítja a minimális frekvenciaértéket (0–32 767), ha a DIR meg van határozva
SV beállítja a pásztázás (sweep) növekményének értékét (0–32 767)
WF hullámforma (0–3)
0 = háromszög
1 = fűrészfog
2 = négyszöghullám (impulzus)
3 = fehér zaj
PW beállítja az impulzusszélességet

A DIR, MIN, SV, WF és PW paraméterek opcionálisak.

A SOUND utasítás 1. paramétere (VC) kiválasztja a lejátszandó hangot.

A 2. paraméter (FREQ) a hang frekvenciáját szabályozza, amely 0-tól 65 535-ig terjedhet.

A 3. (DUR) a hangzás időtartamát szabja meg (60-as másodpercekben). Ha egy hangot egy másodpercig akarunk játszatni, a DUR értéke 60 legyen, mivel 60*1/60 = 1. Ha két másodpercig, akkor 120 legyen a DUR értéke. 10 másodperces időértéknél a DUR 600 stb.
A 4. paraméterrel (DIR) a hang frekvenciájának növelését, ill. csökkenését választjuk meg. Ezt nevezzük *hangpásztázásnak.*

Az 5. paraméter (MIN) azt a minimum frekvenciát jelöli, ahol a pásztázás elkezdődik.

A 6. (SV) a pásztázás növekmény értéke. Hasonlít a FOR...NEXT ciklus növekmény értékéhez. Ha a SOUND utasításban megadjuk a DIR, a MIN és az SV értékeket, a hang először a PREQ paraméterrel meghatározott eredeti szinten szólal meg. Utána a szintetizátor végigpásztázza és lejátssza a frekvenciaértékek teljes tartományának minden egyes szintjét a MIN frekvenciával kezdődően. A növekmény értékét az SV adja meg, és a hang a DIR paraméterrel megadott irányban (magasabban vagy mélyebben) az új szinten hangzik fel.

A 7. paraméter (WF) a hang hullámformáját jelöli. (A hullámformákról részletesen l. a Bonyolult hang- és zenei hatások C128-as üzemmódban c. részt.)
A SOUND utasítás utolsó paramétere a négyszöghullám impulzusának szélességét adja meg, amennyiben azt választottuk hullámformának (l. a Bonyolult hang- és zenei hatások C128-as üzemmódban c. részt.)

Írjunk hangprogramot

Itt az ideje, hogy megírja első hangprogramját. Egy példa a SOUND utasításra:

```
10 VOL 5
20 SOUND 1, 4096, 60
```

Futtassa ezt a programot. A számítógép rövid, magas „bip" hangot ad. A SOUND utasítás előtt először meg kell határoznia a hangerőt, tehát a 10-es sor a hangchip hangerejét állítja be. A 20-as sor az 1. hangot (szólamot) játssza 4096-os frekvenciával 1 másodpercig (60*1/60). Váltson frekvenciát a következő utasítással:

```
30 SOUND 1, 8192, 60
```

Figyelje meg, hogy a 30-as sor magasabb hangot ad, mint a 20-as sor. Így tapasztalhatta a frekvenciabeállítás és a megszólaló hang frekvenciája közötti közvetlen kapcsolatot. Ahogy növeljük a frekvenciát, a számítógép növeli a hangmagasságot. Próbálja ki most ezt az utasítást:

```
40 SOUND 1, 0, 60
```

amelyből kiderül, hogy a 0 értékű frekvencia a legalacsonyabb (annyira, hogy nem is hallható). A 65 535-ös frekvenciaérték szólaltatja meg a lehető legmagasabb hangot.
Helyezze most a SOUND utasítást egy FOR... NEXT ciklusba. Ekkor a cikluson belül egy egész sor frekvenciát szólaltathat meg. Adja hozzá a programhoz a következő utasításokat:

```
50 FOR I=1 TO 65535 STEP 100
60 SOUND 1, I, 1
70 NEXT
```

Ez a programrész a négyszög hullámformát játssza 100-as ugrásokban a legalacsonyabb hangtól a legmagasabbig. Ha külön nem adja meg a hullámformát, a számítógép az alaphullámformát, a szögletest választja (2).
Most változtassa meg a hullámformát: a 60-as sor helyett írja be a következőket, és így futtassa a programot:

```
60 SOUND 1, I, 1, 0, 0, 0, 0, 0
```

A program most az 1. szólamot játssza háromszög hullámformában 1-től 65 353-ig terjedő frekvenciaszámmal, 100-as ugrásokban. Úgy hangzik, mint a népszerű számítógépes játékok egy hanghatása. Most próbálja ki az 1. hullámformát, a fűrészfogat:

```
60 SOUND 1, I, 1, 0, 0, 0, 1, 0
```

A fűrészfog hullámforma hasonlít a háromszöghöz, csak kevesebb a zörej benne. Végezetül hallgassa meg a fehér zaj hullámformát:

```
60 SOUND 1, I, 1, 0, 0, 0, 3, 0
```

Most a programciklus a fehér zaj generátort játszatja a teljes frekvenciatartományon. Először mély, morajló hangot hallunk, majd a frekvencia növelésével nő a hangmagasság is, és olyan lesz, mint egy kilőtt rakéta hangja. Figyelje meg, hogy mind ez idáig nem határoztuk meg a SOUND utasítás összes paramétereit. Vegyük pl. a 60-as sort:

```
60 SOUND 1, I, 1, 0, 0, 0, 3, 0
```

Az 1, I, 1-et követő három nulla a pásztázás paraméterekre vonatkozik. Miután egyik sincsen meghatározva, a pásztázás elmarad. Adja hozzá a következő sort a programhoz:

```
100 SOUND 1, 49152, 240, 1, 0, 100, 1, 0
```

szólam
frekvencia
időtartam
a pásztázás iránya
min. frekvencia
a növekmény értéke
hullámforma
az impulzus szélessége
a szögletes hullámformánál

A 100-as sor a 49 152-es frekvenciánál kezd és 100-asával csökkenti a pásztázást, amig el nem éri a 0 frekvenciaértéket. Az 1. szólam fűrészfog hullámformában (1) minden hangot 4 másodpercig játszik (240*1/60 s). A 100-as sor úgy hangzik, mint egy lehulló bomba, hasonlít a lövöldözős játék hangjához.
Most változtassa meg a 100-as sor néhány paraméterét. Pl. a pásztázás irányát állítsa be 2-re (oszcillál), adjon 32 768 minimális frekvenciaértéket, és növelje a lépés (növekmény) értékét 3000-re. Az új SOUND parancs a következőképpen alakul:

```
110 SOUND 1, 49152, 240, 2, 32768, 3000, 1
```

A 110-es sor nyomán most szirénázó hangot hallunk, mintha a rendőrség épp a nyomunkban volna. Ha kellemesebb hangot szeretne, írja be a következőt:

110 SOUND 1, 65535, 250, 0, 32768, 3000, 2, 2600

Ez a hang egy űrkorszakban játszódó tv-műsorra emlékeztet, mintha az űrhajó legénysége épp tüzet nyitott volna a mit sem sejtő idegenekre.

Mostanáig csak egy szólamban programozott. Érdekes hanghatásokat érhet el a SOUND utasítással, ha több szólamot is használ. Kísérletezzen, és készítsen egy olyan programot, amelyben mind a három szólamot felhasználja. A következő mintaprogram segít a Commodore 128-as szintetizátor chipjének programozásában. A futtatott program bekéri az egyes paramétereket, majd lejátssza a hangot. A következő programot írja be a számítógépbe és futtassa:

A mintaprogram rövid magyarázata:

nem, nyomja meg az N billentyűt, és a program a 130-as sorba ugrik, és lejátssza a hangot. Ha meg akarja adni az opcionális paramétereket, nyomja meg az I billentyűt, és a program a 80-as sorban folytatódik. A 80–110-es sor megadja a pásztázás irányát, a minimum frekvenciát, a növekmény értékét és a hullámformát. A 120-as sor csak akkor írja be a négyszög hullámforma impulzusszélességét, ha a 2 (négyszög) hullámformát választottuk. Végül a 130-as sor lejátssza a hangot a program során megadott paramétereknek megfelelően.

A 140-es sor megkérdezi, hogy akarjuk-e még egyszer hallani a hangot. Ha igen, nyomjuk meg az I billentyűt, ha nem, az N-t. A 150-es sor ellenőrzi, hogy az I billentyűt nyomtuk-e meg. Ha igen, a vezérlés visszamegy a 130-as sorba, és a hang ismét hallható lesz. Ha nem, nyomja meg az I billentyűt, a program a 160-as sorban folytatódik, amely visszaadja a vezérlést a 10-es sornak és az egész program megismétlődik. Ha meg akarja állítani a hangprogramot, nyomja meg egyidejűleg a RUN/STOP és a RESTORE billentyűket.

```
5 REM **** P.10 ****
6 PRINT"␣"
10 PRINT"                    HANGJATEK":PRINT:PRINT:PRINT
20 PRINT"  HANG ADATAINAK MEGADASA":PRINT:PRINT
30 INPUT "SZOLAM  (1-3) ";V
40 INPUT "FREKVENCIA (0-65535) ";F
50 INPUT "IDOTARTAM (0-32767) ";D:PRINT
60 INPUT "MEGAD SPEC. PARAMETEREKET? (I/N) ";B$:PRINT
70 IF B$="N" THEN 130
80 INPUT "SWEEP ALLIT. 0=FEL,1=LE, 2=OSZCILL.";DIR
90 INPUT "MIN. SWEEP FREKVENCIA (0-65535) ";M
100 INPUT "SWEEP STEP VALUE (0-32767) ";S
110 PRINT "HULLAMFORMA   (0=HAROMSZOG)"
111 PRINT "              (1=FURESZFOG)"
112 PRINT "              (2=VAR PUL   )"
113 INPUT "              (3=ZAJ       ) ";W
120 IF W=2 THEN INPUT"ISMETLES SZAMA (0-4095) ";P
130 SOUND V,F,D,DIR,M,S,W,P
140 INPUT "KIVANJA UJBOL HALLANI?  (I/N)";A$
150 IF A$="I" THEN130
160 GOTO  6
```

A 10-es és a 20-es sor kiírja a képernyőre a bevezető üzeneteket. A 30–50-es sor megadja a szólam, a frekvencia és az időtartam paramétereit. A 60-as sor megkérdezi, hogy be akarjuk-e vinni az opcionális paramétereket, pl. a pásztázás (sweep) beállítását és a hullámformát. Ha

Véletlenhangok

A következő program véletlenhangokat generál az RND függvény segítségével. A számítógép minden hangparamétert véletlenszerűen generál. Írja be a programot a számítógépbe, mentse

ki és futtassa. A következő program azt mutatja be, hogy hány ezer hangot képezhet a SOUND utasítás paramétereinek meghatározásával. Íme a lista:

```
5 REM **** P.11 ****
10 PRINT"■VC  FREQ  DIR  MIN  SV  WF   PW "
20 PRINT"-----------------------------------"
30 V=INT(RND(1)*3)+1: REM SZOLAM
40 F=INT(RND(1)*65535): REM FREKVENCIA
50 D=INT(RND(1)*32767): REM IDOTARTAM
60 DIR=INT(RND(1)*3): REM LEPES IRANYA
70 M=INT(RND(1)*65535):REM MIN. FREKVENCIA
80 S=INT(RND(1)*32767):REM LEPES ERTEKE
90 W=INT(RND(1)*4):REM HULLAMFORMA
100 P=INT(RND(1)*4095):REM IMPULZUSSZELESSEG
110 PRINTV; F; DIR; M; S; W; P:PRINT:PRINT
120 SOUND V,F,D,DIR,M,S,W,P
130 SLEEP 4
140 SOUND V,0,0,DIR,0,0,W,P
150 GOTO 10
```

A 10-es és a 20-as sor kiírja a paraméter oszlopok fejlécét és az aláhúzó vonalat. A 3–100-as sor kiszámítja az egyes paramétereket a kijelölt tartományon belül. Pl. a 30-as sor azt számítja ki, hányadik szólam hangozzon fel:

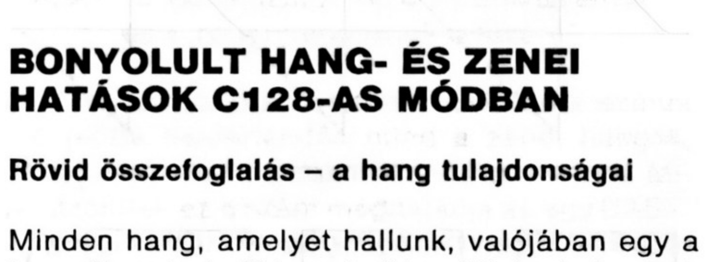
30 V = INT(RND(1)*3) + 1

Az RND a véletlenszám kiindulási értékét határozza meg. A kiindulási érték a számítógép által generált alapszám. Az 1-es közli a számítógéppel, hogy generáljon egy új kiindulási értéket, valahányszor a parancsot végre kell hajtani. Mivel a Commodore 128-asnak 3 szólama van, a 3 azt közli a géppel, hogy 0-tól 3-ig generáljon véletlenszámot. Valószínűleg felfigyelt azonban arra, hogy nincs 0 szólam, tehát a 30-as sorban a + 1 arra utasítja a gépet, hogy 1-től 3-ig generáljon véletlenszámot. Egy bizonyos tartományban véletlenszám előállítása azt jelenti, hogy az adott véletlenszámot a paraméter maximális értékével szorozzuk, jelen esetben 3-mal. Ha a paraméter minimális értéke nagyobb, mint 0, adjunk hozzá a véletlenszámhoz egy értéket, amely a generálandó számok tartományának minimális értékét jelzi (ez esetben 1). Pl. a 40-es sor 0 és 65 535 között állít elő véletlenszámokat. Mivel ez esetben a minimális érték nulla, nem kell az előállított véletlenszámhoz értéket hozzáadni.
A 110-es sor kiírja a paraméterek értékeit. A 120-as sor lejátssza a 30–100-as sorban előállított véletlenszámok által meghatározott hangot. A 130-as sor 4 másodperccel késlelteti a programot, amíg a hang szól. A 140-es sor a 4 másodperces késleltetés után kikapcsolja a hangot. Minden ebben a programban generált hang ugyanannyi ideig szól, mert a 4 másodperc után a 140-es sor kikapcsolja őket. Végezetül a 150-es sor visszaadja a vezérlést a 10-es sornak, és a folyamat addig ismétlődik, amíg a RUN/STOP és a RESTORE billentyűket egyszerre le nem nyomjuk.
Eddig csak a SOUND utasítást használtuk a mintaprogramban. Bár a SOUND-dal lehet kottát is játszani, mégis, a gyors és könnyű hanghatások elérésére a legalkalmasabb. A Commodore 128-as számítógépnek dallamok lejátszására van még néhány más utasítása is. A következő rész azokat a hang- és zenei utasításokat ismerteti, amelyek segítségével bonyolultabb zeneműveket is lejátszhatunk.

BONYOLULT HANG- ÉS ZENEI HATÁSOK C128-AS MÓDBAN

Rövid összefoglalás – a hang tulajdonságai

Minden hang, amelyet hallunk, valójában egy a levegőben terjedő hanghullám. Mint minden hullámot, a hang- (szinusz) hullámot is lehet grafikusan és matematikailag ábrázolni (7.1. ábra).

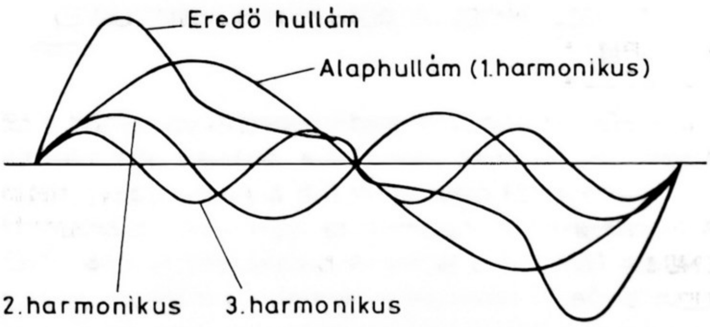


7.1. ábra. Szinuszhullám

A hanghullám egy bizonyos frekvencián rezeg, ez határozza meg a hang magasságát.

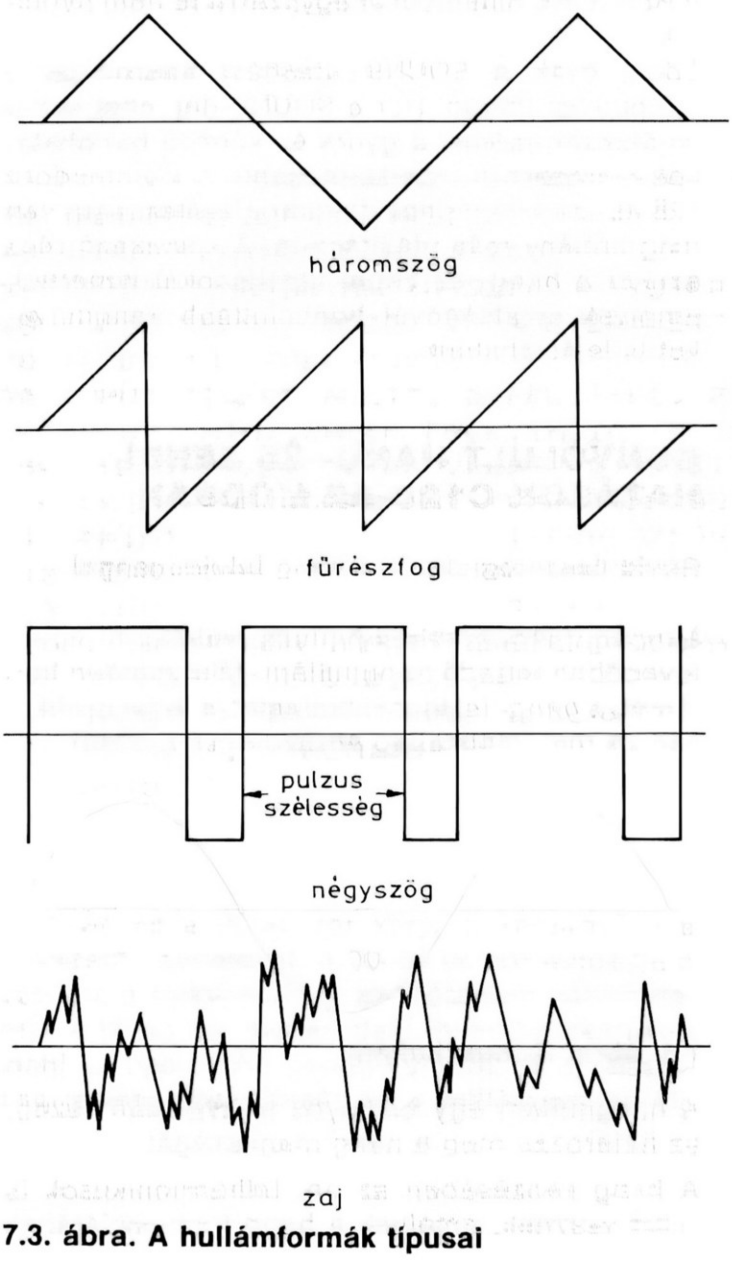
A hang képzésében az ún. felharmonikusok is részt vesznek, amelyek a hang frekvenciájának

egész számú többszörösei. E harmonikus hanghullámok együttese adja a hang minőségét, azaz a hangszínt. A 7.2. ábra az alapfrekvenciák és a felharmonikusok viszonyát mutatja be.



7.2. ábra. A frekvencia és a felharmonikusok

A zenei hang színét a hang hullámformája határozza meg. A Commodore 128-as számítógép négyfajta hullámformát tud generálni: háromszög, fűrészfog, négyszög (impulzus) és zaj, ezeket a 7.3. ábra mutatja.

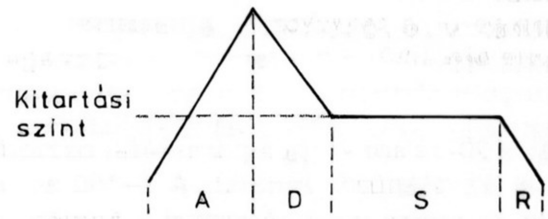


7.3. ábra. A hullámformák típusai

Zenélés a Commodore 128-assal

Az ENVELOPE utasítás

A hang ereje a hangzás időtartama alatt is változik, attól kezdve, hogy először megszólalt, egészen addig, amig már nem hallható többé. Ezek a hangerő-tulajdonságok: a felfutás, lecsengés, kitartás és elengedés (ADSR – attack, decay, sustain, release). A *felfutási idő* az, ami alatt a hangerő eléri a csúcsértéket, a *lecsengési idő*, ami alatt leesik a középértékre (kitartási szint). A *kitartási szinten* közepes hangerővel halljuk a hangot. Az *elengedési idő* az az idő, ami alatt a hang a kitartási szintről a nullára esik. A burkológörbe (ENVELOPE) generátor vezérli a hang ADSR paramétereit. A 7.4. ábra a burkológörbe grafikus képét mutatja.



7.4. ábra. ADSR fázisok

A Commodore 128-as minden ADSR paramétert 16 különböző értékben tud alkalmazni. Így a keletkező hang tulajdonságait maximálisan tudjuk befolyásolni.

A Commodore 128-as hangutasításai közül az egyik leghatékonyabb az ENVELOPE utasítás, amellyel az ADSR görbét és a hullámformát lehet meghatározni. Ez az utasítás állítja be a szintetizátor chipben a különféle vezérléseket, hogy minden hang egyedi legyen. Az ENVELOPE utasítással lehet kezelni a SID szintetizátort és kiválasztani a hang- és zenei hatásokhoz szükséges speciális ADSR beállításokat, valamint a hullámformát. Az ENVELOPE utasítás formátuma a következő:

ENVELOPE e[,a[,d[,s[,r[,wf[,Pw]]]]]]

ahol
e a burkológörbe száma (0–9);
a a felfutás (0–15);
d a lecsengés (0–15);
s a kitartás (0–15);
r az elengedés (0–15);
wf a hullámforma: 0 – háromszög,
1 – fűrészfog,
2 – impulzus (négyszög),
3 – zaj,
4 – gyűrűsmodulátor;
pw az impulzus szélessége.

Az eddig meg nem határozott paraméterek definíciói:

Burkológörbe (Envelope) – a zenei hang tulajdonságai, amelyeket a hullámforma és az ADSR értékek határoznak meg. Pl. a gitárhang ADSR burkológörbéje és hullámformája eltér a zongoráétól.

Hullámforma – a hanghullám típusa, amelyet a hang kísérő-felharmonikusainak kombinációja határoz meg. A felharmonikusok az alaphang frekvenciájának többszörösei. Az egyes hullámformák által generált hangok tulajdonságai felismerhetően különböznek egymástól, l. a 7.3. ábrát.

Impulzusszélesség – az impulzus hullámforma által generált hangok közt eltelt idő hossza.
Most már látja, mi mindent tud az ENVELOPE utasítás. Ez vezérli a szintetizátor által lejátszott hangok legtöbb zenei tulajdonságát. A Commodore 128-as számítógépnek 10 különböző hangszerre 10 előre meghatározott burkológörbéje van. Ha ezeket használja, már nem kell külön megadnia az ADSR paramétereket, a hullámformát és az impulzusszélességet, mindössze a burkológörbe számát. A többi paramétert a Commodore 128-as automatikusan kiválasztja. Íme a különböző hangszerek előre meghatározott értékei:

7.1. táblázat

Az ENVELOPE utasítás alapparaméterei

Burkológörbe száma	Hangszer	Felfutás	Lecsengés	Kitartás	Elengedés	Hullámforma	Szélesség
0	Zongora	0	9	0	0	2	1536
1	Harmonika	12	0	12	0	1	
2	Síplàda	0	0	25	0	0	
3	Dob	0	5	5	0	3	
4	Fuvola	9	4	4	0	0	
5	Gitár	0	9	2	1	1	
6	Csembaló	0	9	0	0	2	512
7	Orgona	0	9	9	0	2	2048
8	Trombita	8	9	4	1	2	512
9	Xylofon	0	9	0	0	0	

Az előbbiek ismeretében írja be a következő utasítást:

```
10 ENVELOPE 0, 5, 9, 2, 2, 2, 1700
```

ez az utasítás újradefiniálja a zongora burkológörbe alapértékeit: felfutás = 5, lecsengés = 9, kitartás = 2, elengedés = 2, a hullámforma ugyanaz marad, és az impulzus hullámforma szélessége 1700. A fentieket csak a PLAY utasítással lehet megvalósítani, ezt majd a későbbiekben tárgyaljuk.
A zenei programozás következő lépése a hang alap hangerejének beállítása:

```
20 VOL 8
```

A VOL (hangerő) utasítással 0 és 15 között lehet a hangerőt beállítani, 15 a maximális, 0 a kikapcsolt érték.

A TEMPO utasítás

A következő lépés a tempó, azaz a dallam ütemének meghatározása. Az utasítás formátuma:

```
TEMPO n
```

ahol n egy 0 és 255 közötti érték. Ha a tempó értékét külön nem határozza meg, a Commodore 128-as automatikusan 8-ra áll be. Tegye hozzá a programhoz a következő utasítást:

```
30 TEMPO 10
```

A PLAY utasítás

Itt az ideje, hogy megtanulja, hogyan lehet a számítógépen dallamot lejátszani. A PRINT utasítást már ismeri, hasonló a PLAY is. A PRINT szöveget ír ki, a PLAY pedig zenei hangokat képez. A PLAY utasítás általános formátuma a következő:

```
PLAY "a szintetizátor vezérlőkaraktereinek
         és a zenei hangoknak a füzére"
```

A PLAY utasításba betehető karakterek száma 255 (ebbe beleértendők mind a zenei hangok, mind pedig a szintetizátor vezérlőkarakterei). Mivel azonban ez a szám meghaladja az egy BASIC 7.0 programsorban megengedett karakterek számát (160), legalább két füzért össze kell adni. Ezt el lehet kerülni, ha ügyelünk arra, hogy a PLAY parancsok ne legyenek hosszabbak 160 karakternél, azaz egy programsornál. (Ez 40 oszlopos módban 4, 80 oszlopos módban pedig 2 képernyősort jelent.) Így gyorsabban és könnyebben használható PLAY parancsfüzéreket lehet készíteni.
Zenei hangok lejátszásához a hang betűformáját kell megadni idézőjelben, pl. a skála hangjait így kell megszólaltatni:

```
40 PLAY "C D E F G A B"
```

(Megjegyzés: a magyarban a C-dúr skála 7. hangja nem B, hanem H, a B ennek a fél hanggal leszállított változata.)

Ez az utasítás a 0 burkológörbe-számú, zongora hangon szólaltatja meg a hangokat. Minden alkalommal, miután futtatta a programot, tartsa lenyomva a RUN/STOP billentyűt és nyomja meg a RESTORE-t, hogy a szintetizátor chipet visszaállítsa alaphelyzetbe.

A hangok időtartamát is meg lehet határozni, ha a következő betűket tesszük az idézőjelben álló hangok elé:

W – egész hang;
H – félhang;
Q – negyedhang;
I – nyolcadhang;
S – tizenhatod hang.

A beállítás alapértéke, ha külön nem adjuk meg, az egész hangokra vonatkozik. Szünetet a PLAY füzérbe iktatott R (Rest = szünet) betűvel lehet játszatni.
Azt is közölhetjük a számítógéppel, hogy várjon, míg minden hang eléri az ütem végét, ha idézőjelben a következőt is belevesszük az utasításba:

M – várj az ütem végéig.

A Commodore 128-as számítógépnek vannak szintetizátorvezérlő karakterei, amelyeket idézőjelben tehetünk a PLAY füzérbe. Így minden hangnak külön vezérlése lesz, és egyetlen füzéren belül is meg lehet változtatni a szintetizátor vezérlését. A következő táblázat megadja a vezérlőkaraktereket és számtartományukat. A karakter után álló n a kívánt számot helyettesíti.

7.2. táblázat

A hangszintetizátor vezérlőkarakterei

Vezérlő-karakter	Jelentés	Tartomány	Alapérték
V n	Szólam	1–3	1
O n	Oktáv	0–6	4
T n	Burkológörbe	0–9	0
U n	Hangerő	0–15	9
X n	Szűrő	0 = ki 1 = be	0

Bár a SID chip a vezérlőkaraktereket bármilyen sorrendben fel tudja dolgozni, jobb eredményt lehet elérni, ha az ábrán megadott sorrendben helyezi el őket a füzérben.
Nem kell mindenképpen megadni a vezérlőkaraktereket, de ajánlatos a szintetizátor lehetőségeit maximálisan kihasználni. A számítógép automatikusan a 7.2. táblázatban megadott alapértékekre áll be; ha nem határozzuk meg a karaktereket, a SID chip csak egy burkológörbét, egy szólamot és egy oktávot tud játszani, szűrés nélkül. Ajánlatos azonban a maximális vezérelhetőség érdekében a füzéren belül meghatározni a karakterek értékét. Ha ENVELOPE utasítást ír, és az alapérték helyett saját beállításokkal dolgozik, a PLAY füzérben az „envelope" (burkológörbe) karakterszámának meg kell egyeznie az ENVELOPE utasításban szereplővel. Ha megelégszik az alapértékekkel, nem kell ENVELOPE utasítást írnia, mindössze a PLAY utasításban a T vezérlőkarakterrel meg kell határoznia a burkológörbe számát.
Most egy PLAY utasítás mintája következik, amely füzérben adja meg a SID chip vezérlőkaraktereit. Tegye hozzá programjához ezt a sort, és figyelje meg a különbséget a 40-es sor PLAY utasítása és e között:

```
50 PLAY "V2 O5 T7 U5 X0 C D E F G A B"
```

Ez az utasítás ugyanazokat a hangokat játssza, mint a 40-es sor, de két szólamban, a hangok egy oktávval magasabban szólalnak meg (5), a hangerő csökkent (5), és a szűrő ki van kapcsolva. Egyelőre a szűrőt ne kapcsolja be, majd ha a következő részben megismerkedik a szűréssel, visszatérhet ehhez a részhez, és megfigyelheti a különbséget. Az 50-es sorban új hangszert választottunk, az orgonát, a T7 vezérlőkarakterrel. Most a program két különböző hangszeren játszik, két külön szólamban. Írja a programhoz a következő sort is:

```
60 PLAY "V3 O6 T6 U7 X0 C D E F G A B"
```

A V3 a harmadik szólam, az O6 ezt egy oktávval magasabbra helyezi, a T6 pedig a csembalót jelenti. Az U7 a hangerőt 7-re állítja be, az X0 pedig mindhárom szólamnál kikapcsolja a szűrőt. Most a program mindhárom szólamot játssza, egy-egy oktávval magasabban, három külön hangszeren: zongorán, orgonán és csembalón.
Eddig csak egész hangokat játszottunk. A hangok hosszúságát a következő karakterekkel lehet meghatározni:

```
70 PLAY "V2 O6 TO U7 X0
H C D Q E F I G A S B"
```

A 70-es sor a 2. szólamot a 6. oktávon, 7-es hangerővel játssza, ismét zongorán, szűrő nélkül. Ez az utasítás a C és D hangokat félhangnak, az E-t és az F-et negyednek, a G-t és az A-t nyolcadnak, a B-t pedig tizenhatodnak játssza. Figyelje meg a 40-es sor zongora hangját és az újradefiniált zongorahangot a 70-es sorban. A 40-es sor hangja jobban hasonlít a zongorára.
A következő karakterek használatával felemelt, leszállított és pontozott hangokat is lehet játszani.

– felemelt (kereszt előjegyzés)
$ – leszállított (♭ zenei leszállítás)
. – pontozott hang

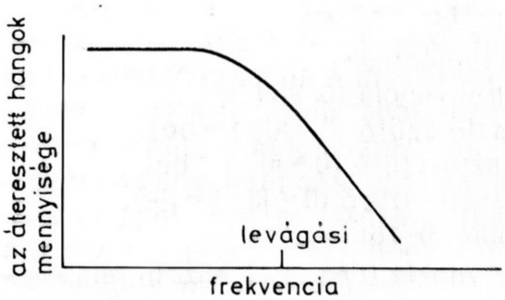
A pontozott hang másfélszer annyi ideig szól, mint a pont nélküli. Most írjuk be a felemelt, leszállított és pontozott hangokat is az utasításba:

```
80 PLAY "V1 O4 T4 U8 X0 .H
C D Q E F I $ G A .S B"
```

A 80-as sor az első szólamot a 4. oktávban, 8-as hangerővel, fuvolahangon játssza, szűrő nélkül. C-t és D-t pontozott félhangnak halljuk, E és F felemelt negyedek, G és A leszállított nyolcadok míg B felemelt, pontozott tizenhatod. A PLAY füzérbe bárhová elhelyezhet szüneteket is (R). Nem szükséges üres karakterhelyeket hagyni a füzérben, mindössze a könnyebb olvashatóság érdekében írtuk így.
Mostanáig a példaként bemutatott utasításokban a szintetizátor szűrője nem volt bekapcsolva, így hatását sem figyelhettük meg. Most, hogy már a legtöbb hang- és zenei utasítással valamint a SID vezérlőkarakterekkel is megismerkedett, a következő részből megtudhatja, hogyan lehet javítani a zenei minőségét a FILTER (szűrő) utasítással.

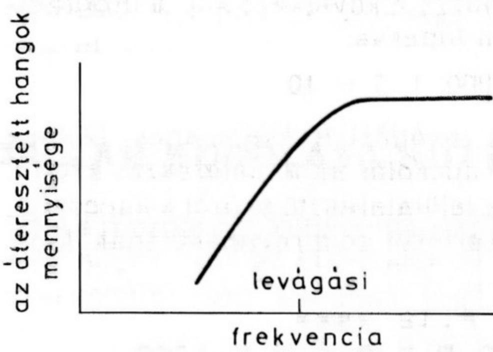
A SID szűrő

Ha már az ENVELOPE, az ADSR, a VOL és a TEMPO utasítások értékeit meghatározta, szintetizált hangjai tökéletesítésére használja a FILTER utasítást is. A programban a FILTER utasítás megelőzi a PLAY-t, ajánlatos azonban először a hang előállítását alaposan elsajátítani, és csak azután foglalkozzon a szűréssel. Mivel a SID chipnek csak egy szűrője van, mindhárom szólamhoz ezt alkalmazzuk. Le lehet játszani a dallamokat a szűrő nélkül is, használatával azonban jelentős mértékben javítható a hang minősége. A fejezet elején, A hang tulajdonságai c. részben úgy definiáltuk a hangot, hogy az egy, a levegőben bizonyos sebességgel rezgő hullám. A hullám egy alapfrekvenciából és kísérő felharmonikusokból áll (l. 7.2. ábra). Ezek a felharmonikusok adják a hangszínt, a hangnak a hullámforma által meghatározott tulajdonságait. A SID chip szűrője azt teszi lehetővé, hogy egy hullámforma felharmonikusait hangsúlyozzuk vagy kiküszöböljük, megváltoztatva ezáltal a hangszínt.
A SID chip háromféle módon szűri a hangokat: van aluláteresztő, felüláteresztő és sáváteresztő szűrő. E három módszer együttesen is alkalmazható, erről a következő részben lesz szó. Az aluláteresztő szűrő egy meghatározott szint – a levágási frekvencia – fölött szűri ki a hangokat. A levágási frekvenciaszint a határvonal, amely jelzi, hogy melyik frekvenciaszint lesz hallható és melyik nem. Az aluláteresztő szűrő alkalmazása esetén a SID chip a levágási frekvenciaszint alatt minden frekvenciát lejátszik, míg az a fölöttieket kiszűri. Mint ahogy a neve is utal rá, az alacsony frekvenciákat a szűrő átengedi, a magasakat nem. Az aluláteresztő szűrő telt, zengő hangokat eredményez (7.5. ábra).



7.5. ábra. Aluláteresztő szűrő

A felüláteresztő szűrő, ezzel ellentétben, a levágási szint feletti frekvenciákat engedi át. Mindent, ami alatta van, kiszűr (7.6. ábra). A felüláteresztő szűrő fémes, kongó hangokat ad.



7.6. ábra. Felüláteresztő szűrő

A sáváteresztő szűrő a levágási szint közvetlen közelében enged át frekvenciákat, míg a levágási szintet körülvevő sáv fölött és alatt a frekvenciákat kiszűri (7.7. ábra).



7.7. ábra. Sáváteresztő szűrő

A FILTER utasítás

A FILTER utasítás meghatározza a levágási szintet, a szűrő típusát és a rezonanciát. A rezonancia az a hanghatás, amikor a hanghullám frekvenciája elhal, azaz a levágási frekvenciához közelít. A rezonancia a hang élességét és tisztaságát határozza meg: minél magasabb a rezonancia, annál élesebb a hang. A FILTER utasítás formátuma a következő:

FILTER cf, lp, bp, hp, res

A paraméterek jelentése:
cf – levágási frekvencia (0–2047);
lp – aluláteresztő szűrő (0 = ki, 1 = be);
bp – sáváteresztő szűrő (0 = ki, 1 = be);
hp – felüláteresztő szűrő (0 = ki, 1 = be);
res – rezonancia (0–15).

A levágási frekvencia 0 és 2047 közötti tetszőleges érték. Ha be akarja kapcsolni az aluláteresztő szűrőt, adjon a FILTER utasítás második paraméterének 1 értéket. Ha a sáváteresztő szűrőt kívánja bekapcsolni, a harmadik paraméter legyen 1, míg a felüláteresztő szűrő esetében a negyedik. Ha a három közül bármelyiknek 0 értéket ad, a megfelelő szűrő kikapcsol. Egyet, kettőt vagy mind a hármat is kikapcsolhatja egyszerre. Most, hogy a FILTER utasításról már van némi fogalma, adja hozzá a következő sort a programhoz, de még ne futtassa:

45 FILTER 1200, 1, 0, 0, 10

A 45-ös sor a levágási frekvenciának 1200-as értéket ad, bekapcsolja az aluláteresztő szűrőt, míg a sáv- és a felüláteresztő szűrőt kikapcsolja, valamint 10-es értéket ad a rezonanciának. Most menjen vissza a programban, és az eddigi PLAY utasításokban kapcsolja be a szűrőt oly módon, hogy valamennyi szűrő vezérlőkaraktert X0 helyett X1-re változtatja. Helyezze kiindulási helyzetbe a hangchipet a RUN/STOP és a RESTORE billentyűkkel és futtassa (RUN) újra a hangprogramot. Figyelje meg, milyen más a hangzás a szűrővel. Cserélje ki a 45-ös sort:

45 FILTER 1200, 0, 1, 0, 10

Az új 45-ös sor kikapcsolja az aluláteresztő szűrőt, és bekapcsolja a sáváteresztő szűrőt. A RUN/STOP és a RESTORE billentyűk lenyomása után futtassa újra a programot. Figyelje meg az aluláteresztő és a sáváteresztő szűrő közti különbséget. Ismét változtassa meg a program 45-ös sorát:

45 FILTER 1200, 0, 0, 1, 10

Helyezze ismét kiindulási helyzetbe a hangchipet és futtassa a programot, majd figyelje meg a felüláteresztő szűrő működését az aluláteresztő és a sáváteresztő szűrőhöz képest. Kísérletezzen különböző levágási frekvenciákkal, rezonanciaszintekkel és szűrőkkel a zenei hangzás tökéletesebbé tétele érdekében.

A zenei program összeállítása

Készen van az első zenei programja. Most már kedvenc dalait is beprogramozhatja. Fűzze össze a program alkotóelemeit a következő programlista szerint. Ne aggódjon, ez ugyanaz a program, mint amit az előbb fölépített, csak PRINT utasításokat adtunk hozzá, hogy tudja, melyik programsort játssza épp a számítógép.

```
5 REM **** P.12 ****
10 ENVELOPE 0,5,9,2,2,2,1700
15 VOL 8
20 TEMPO 10
25 PRINT"30. SOR "
30 PLAY"C D E F G A B M"
35 FILTER 1200,0,0,1,10
40 PRINT"45. SOR - SZURO KI"
45 PLAY"V2 O5 T7 U5 X0 C D E F G A B M"
50 PRINT"    SZURO BE"
55 PLAY "V2 O5 T7 U5 X1 C D E F G A B M"
60 PRINT"65. SOR - SZURO KI"
65 PLAY "V3 O6 U7 T6 X0 C D E F G A B M"
70 PRINT"    SZURO BE"
75 PLAY"V3 O6 U7 T6 X1 C D E F G A B M"
80 PRINT "85. SOR - SZURO KI"
85 PLAY"V2 O6 T0 U7 X0 H CD Q EF I GA S B M"
90 PRINT"    SZURO BE"
95 PLAY"V2 O6 T0 U7 X1 H CD Q EF I GA S B M"
100 PRINT"105. SOR - SZURO KI"
105 PLAY"V1 O4 T4 U8 X0 H .C D Q # EF I $ GA S .B M"
110 PRINT"    SZURO BE"
115 PLAY"V1 O4 T4 U8 X1 H .C D Q # EF I $ GA S .B M"
```

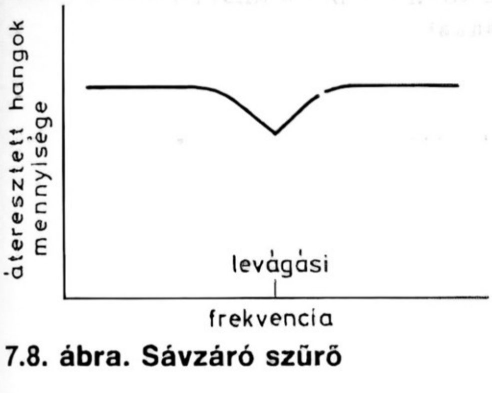
A 10-es sorban az ENVELOPE utasítás a zongora burkológörbéjét határozza meg, ahol a felfutás 0, a lecsengés 9, a kitartás 0, míg az elengedés 0. 1700-as impulzusszélességgel négyszög hullámformát választottunk. A 15-ös sor a hangerőnek (VOLume) 8-as értéket ad. A 20-as sorban a TEMPO értéke 10.

A 35-ös sor a 30–115-ös sorokban játszott hangokat szűri. A levágási frekvencia 1200. Ezen kívül, ebben a sorban kapcsoljuk ki az aluláteresztő és a sáváteresztő szűrőket is, a levágási frekvenciát követő két 0 értékkel, amelyek után az 1-es bekapcsolja a felüláteresztő szűrőt.

A rezonancia értéke 10, ezt a FILTER utasítás utolsó paramétere jelzi. A 30-as sor lejátssza a C, D, E, F, G, A, B hangokat, ebben a sorrendben. A 45-ös sor ugyanezeket a hangokat szólaltatja meg, de a SID vezérlőkaraktereket a következőképpen határozza meg. U5 – 5-ös hangerő, V1 – 1. szólam és O5 – 5. oktáv. Ne feledje, hogy a SID vezérlőkarakterekkel egy füzéren belül meg lehet változtatni a szintetizátor vezérlését, így maximális vezérlést és ellenőrzést biztosít. A 65-ös sorban az U7 7-es hangerőt jelent, a V3 a szólamra, a O6 a 6. oktávra vonatkozik, míg az X0 kikapcsolja a szűrőt. A 65-ös sor ugyanazokat a hangokat játssza, mint a 30-as és a 45-ös, csak más hangerővel, szólamban és oktávon. A 85-ös sorban a 65-öshöz képest nem változik a hangerő, a szólam és az oktáv, azonban a C és a D félhang hosszúsággal szólal meg, az E és az F negyed, a G és az A nyolcad, míg a B tizenhatod hang hosszúságú. A 105-ös sorban 7-es a hangerő, a szólam 1-es, az oktáv a 4. és a szűrő ki van kapcsolva. Ugyanakkor a C pontozott félhang, az E felemelt negyed, a G és az A leszállított nyolcadok, a B pedig pontozott, felemelt tizenhatod.

Magas fokú szűrés

Az előző FILTER utasítások mindegyike csak egyetlen szűrőt használt. Ha különböző szűrési hatásokat akar elérni, kombinálhatja is a SID chip három szűrőjét. Pl. az aluláteresztő és a felüláteresztő szűrő együttes használata adja a sávzáró szűrőt. Ez a levágási szint fölött és alatt átengedi a frekvenciákat, míg a levágási szinthez közelieket kiszűri. A sávzáró szűrő grafikus képe a 7.8. ábrán látható.



7.8. ábra. Sávzáró szűrő

Érdekes hatásokat lehet elérni a sáváteresztő és az alul- vagy felüláteresztő szűrő együttes alkalmazásával. A sáváteresztő szűrő és az aluláteresztő szűrő együtt a levágási szint alatt engedi át a frekvenciákat, a többit kiszűri. A sáváteresztő és a felüláteresztő szűrő együttes alkalmazása a levágási szint felett engedi át a frekvenciákat, a levágási szint alatt minden hangot kiszűr.

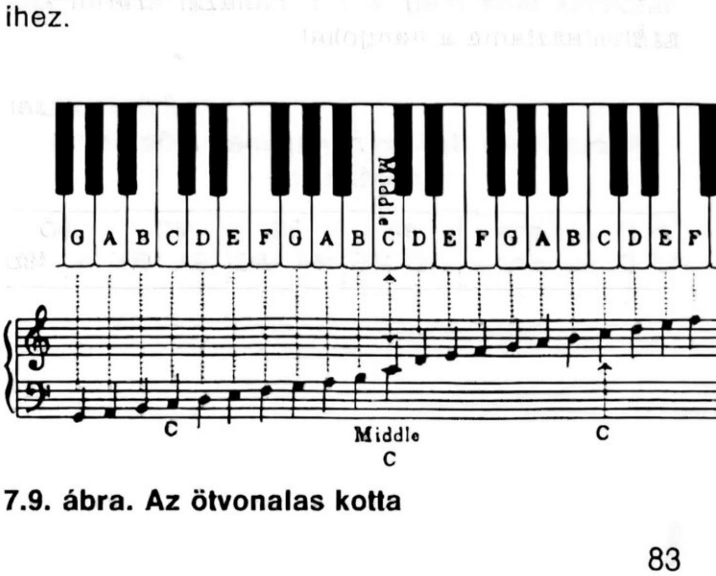
Kísérletezzen a szűrők kombinációjával, hogy lássa a zenei hangok és hanghatások kiemelésének különböző módjait. A szűrőknek az a rendeltetésük, hogy a SID chip többi alkotóeleme által létrehozott hangokat tökéletesítsék. Ha már kiválasztotta a zenei hangokat vagy hangeffektusokat a SID chippel, menjen vissza a programban, és adja hozzá a szűrőhatást, hogy a lehető legtisztább hangok szólaljanak meg.

Most már mindent tud ahhoz, hogy Commodore 128 BASIC-ben saját zenei programokat írjon. Kísérletezzen a különböző hullámformákkal, ADSR beállításokkal, tempókkal és szűrőkkel. Kottából nézzen ki egy skálát és a füzérbe írja be a hangokat egymás után. A SID vezérlőkarakterekkel adjon hangsúlyt bizonyos hangoknak. A Commodore 128-as szintetizátorát együttesen alkalmazhatja a Commodore 128-as grafikai utasításokkal, így hangos videókat, ill. animációs filmeket is készíthet.

DALLAM KÓDOLÁSA KOTTÁBÓL
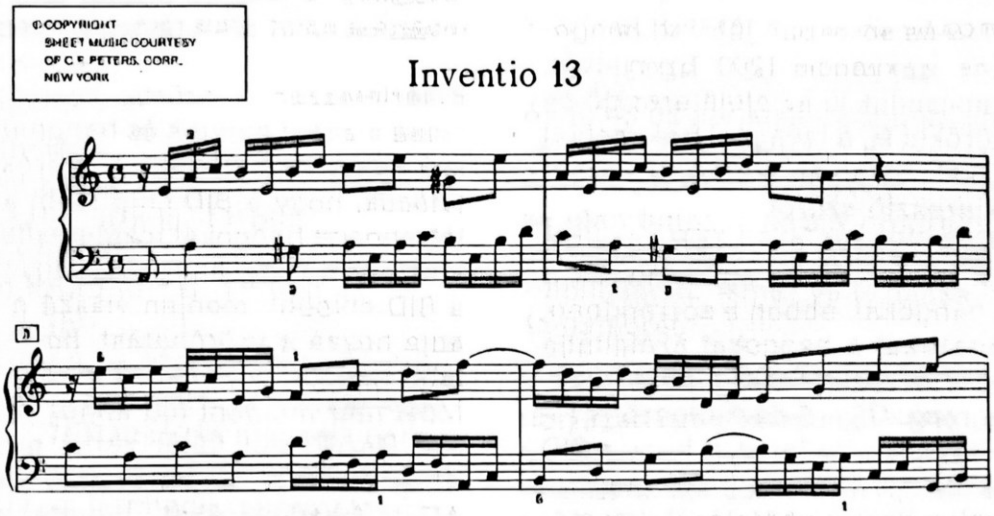
Ebből a részből egy kottaminta alapján megtanulhatja, hogyan kell az öt vonalon ábrázolt hangokat bekódolni, azaz lefordítani a számítógép nyelvére.

Mindez lényegesen könnyebben és gyorsabban megy, ha tud kottát olvasni. Nem kell azonban zenésznek lennie ahhoz, hogy a Commodore 128-ason zenét tudjon lejátszani. Azok számára, akik nem tudnak kottát olvasni, a 7.9. ábra megmutatja, hogy néz ki az öt vonal, és az ott ábrázolt hangok hogyan viszonyulnak a zongora billentyűihez.



7.9. ábra. Az ötvonalas kotta

A 7.10. ábra Johann Sebastian Bach 13. Invenciójának részlete. Bár a művet Bach néhány száz évvel ezelőtt írta, egy olyan korszerű szintetizátoron is lejátszható és élvezhető, mint a Commodore 128-as SID chipje. Íme a 13. Invenció néhány bevezető üteme:



7.10. ábra. Bach 13. Invenciójának részlete

Mivel a zeneműveknél az idő és az együtthangzás alapvető fontosságú, pontosan ellenőrizze a két szólam hangjait. A 7.3. táblázatban az első sor első hangja nyolcad hosszúságú A hang. A 2. szólam első két hangja pedig tizenhatod hosszúságú D és E. Ebben az esetben az 1. szólam nyolcadhangját kell először beírnia a PLAY füzérbe, majd közvetlenül utána a 2. szólam tizenhatodjait. Hogy a példát tovább folytassuk: a 7.3. táblázat 1. szólamának második hangja nyolcad B. Ez időben egyenlő a két tizenhatoddal, F-fel és G-vel a 2. szólamban. Az ütem betartása érdekében írja be a nyolcad B-t az 1. szólam füzérébe, majd közvetlenül utána a 2. szólamba a két tizenhatodot, F-et és G-t.

Legjobb úgy kezdeni a dallam kódolását, hogy a hangokat egy közvetítő kódra írjuk át. Írja le a felső szólam hangjait egy papírra, majd utána az alsó szólam hangjait is. A hangértékek elé kódolja be a hangzás időtartamát is. Pl. a nyolcadhangot előzze meg egy 8-as, a tizenhatodot a 16-os stb. Most úgy válassza szét a hangokat, hogy a felső szólam egy ütembe eső hangjai arányosak legyenek időben az alsó szólam egy ütemével.
Ha a zeneműnek harmadik szólama is van, úgy ez is a két felső szólammal időarányosan legyen bekódolva. Miután minden sor hangjait szétválasztottuk, külön szólam (V1) fogja lejátszani az első szólam hangjait, a másik (V2) a másodikét, és a V3 szólam a harmadikét (amennyiben létezik).
Mondjuk, hogy a felső szólam négy nyolcadhanggal kezdődik, az alsó pedig nyolc tizenhatoddal. Mivel időarányosan egy nyolcadhang két tizenhatodnak felel meg, a 7.3. táblázat szerint kell szétválasztania a hangokat.

7.3. táblázat
Kétszólamú dallam hangjainak időarányos ábrázolása

V1 (1. szólam)	8A	8B	8C	8D
V2 (2. szólam)	16D 16E	16F 16G	16A 16B	16C 16D

Általában mindig a hosszabb időtartamú hanggal kezdje. Pl. ha egy ütem két tizenhatoddal kezdődik az alsó szólamban, a felső pedig egy nyolcaddal, akkor először a nyolcadhangot írja be a füzérbe, mert annak szólnia kell, amíg a számítógép kikeresi a két tizenhatodot. Időt kell hagyni a gépnek, hogy először a hosszabb hangot játssza, majd utána a rövidebbet, mert különben a dallam nem lesz időarányos.
A következő mintaprogram a 13. Invenciót játssza. Írja be a számítógépbe, mentse ki, hogy később is használni tudja, majd futtassa:

Az előző részben ismertetett technikával kódolhatja kedvenc kottáit, majd lejátszathatja a Commodore 128-assal.

****

```
5 REM **** P.13 ****
10 REM  J S BACH  13. INVENTIO
20 TEMPO 6
30 PLAY "V104T7U8X0": REM 1 SZOLAM=ORGONA
40 PLAY "V204T7U8X0": REM 2 SZOLAM=ZONGORA
50 REM ELSO UTEM
60 A$="V201IAV103IEV202QAV103SA04C03BEV202I#GV103SB04DV104ICV202SAEM"
70 B$="V104IEV202SA03CV103I#GV202SBEV104IEV202SB03D"
80 REM MASODIK UTEM
90 C$="V203ICV103SAEV202IAV103SA04CV202I#GV103SBEV202IEV103SB04D"
100 D$="V104ICV202SAEV103IAV202SA03CV104QRV202SBEB03D"
110 REM HARMADIK UTEM
120 E$="V203ICV104SREV202IAV104SCEV203ICV103SA04CV202IAV102SEG"
130 F$="V103IFV203SD02AV103IAV202SFAV104IDV202SDFV104IFV201SA02C"
140 REM NEGYEDIK UTEM
150 G$="V201IBV104SFDV202IDV103SB04DV202IGV103SGBV202IBV103SDF"
160 H$="V103IEV202SGEV103IGV202SEGV104ICV202SCEV104IEV201SGB"
170 REM OTODIK UTEM
180 I$="V201IAV104SECV202ICV103SA04CV103IFV202SDFV104IDV201SB02D"
190 J$="V201IGV103SDBV201IBV103SGBV103IEV202SCEV104ICV201SA02C"
200 REM HATODIK UTEM
210 K$="V201IFV104SC03AV201IDV103SFAV103IDV201SG02GV103IBV202SFG"
220 M$="V201IAV104SC03AV202I#FV104SCEV201IBV104SD03BV202I#GV104SDF"
230 REM HETEDIK UTEM
240 N$="V202ICV104SECV202IAV104SEGV202IDV104SFEV202I$BV104SDC"
250 O$="V202I#GVI03SB04CV202IFV104SDEV202IDV104SFDV201IBV104S#GD"
260 REM NYOLCADIK UTEM
270 P$="V202I#GV104SBDV202IAV104SCAV202IDV104SFDV202IEV103SB04D"
280 Q$="V202IFV103S#GBV202I#DV104SC03AV202IEV103SEAV202IEV103SB#G"
290 REM KILENCEDIK UTEM
300 R$="V201HAV103SAECE02QA"
310 PLAY A$:PLAY B$:PLAY C$:PLAY D$:PLAYE$
320 PLAY F$:PLAY G$:PLAY H$:PLAY I$:PLAY J$
330 PLAY K$:PLAY M$:PLAY N$:PLAY O$:PLAY P$
340 PLAY Q$:PLAY R$
```

Mostanára megismerte a C128-as üzemmódban használható leghatékonyabb BASIC 7.0 parancsokat. A következő részben megtanulhatja, hogyan kell kezelni a számítógép 40 és 80 oszlopos képernyőkijelzéseit.

8. ALFEJEZET

80 oszlop használata

BEVEZETÉS

A 40/80 billentyű

Videocsatlakozók és monitorok

A monitor csatlakoztatása 88

Monitortípusok 88

Gyári csomagolású 80 oszlopos szoftver használata 88

80 oszlopos programok készítése 89

40 és 80 oszlop együttes használata 89

BEVEZETÉS

C128-as és CP/M üzemmódban választhat 40 és 80 oszlopos képernyőkijelzés között, sőt egyetlen programon belül mindkettőt használhatja.
Mindkét képernyőméretnek megvan a maga külön haszna. A 40 oszlopos képernyő azonos azzal a mérettel, amelyet a Commodore 64-es használ. 40 oszlopos képernyővel megvalósítható a Commodore 128-as összes grafikus lehetősége. Köröket, grafikonokat, sprite-karaktereket, dobozokat és egyéb formákat rajzolhat nagyfelbontású vagy többszínű módban. Lehetőség nyílik sprite-ok használatára is.
80 oszlopos képernyővel programsoronként kétszer annyi karaktert jeleníthet meg. 80 oszlopos módban használhatja a billentyűket standard grafikus karaktereit és színeit.
Olyan programot is írhat, amelyhez két monitor kell, így mindkét formátum előnyeit élvezheti, mert egy-egy monitor a program különböző részeit külön valósítja meg. Így pl. a szöveget a 80 oszlopos képernyőn írathatja ki, míg a grafikát a 40 oszloposon jelenítheti meg.

A 40/80 BILLENTYŰ

Ez a billentyű a képernyőszélesség beállítására szolgál (40 vagy 80 oszlop). A billentyű megnyomása csak akkor hatásos, ha a következő feltételeknek legalább egyike teljesül:

1. a gép be van kapcsolva;
2. a RESET gomb le van nyomva;
3. a RUN/STOP és a RESTORE billentyűket egyidejűleg lenyomtuk.

A 40/80 billentyű úgy működik, mint a SHIFT/LOCK billentyű, nyomásra zár, és addig nem enged föl, amíg újra meg nem nyomjuk. Ha a billentyű nincs lenyomva, de az előbbi három feltétel egyikét teljesítettük, a képernyő 40 oszlopra van beállítva. Ha a gép bekapcsolása előtt lenyomjuk a billentyűt, ami ezáltal zár, majd a három feltétel közül legalább egyet teljesítünk, a képernyő 80 oszlopra lesz beállítva. Ha a számítógép valamelyik képernyőformátumban fut (40 vagy 80 oszlop), nem lehet a 40/80 billentyűvel a másik formátumba váltani. Ebben az esetben meg kell nyomni, majd elengedni az ESC billentyűt, és lenyomni az X billentyűt.

VIDEOCSATLAKOZÓK ÉS MONITOROK

A monitor csatlakoztatása

Győződjön meg arról, hogy a monitor helyesen van-e csatlakoztatva a számítógép hátán levő bemenetekhez (port). Két aljzat van, az egyik felirata: VIDEO, a másiké RGBI.

A VIDEO feliratú bemenetbe a 40 oszlopos kompozit video monitort kell csatlakoztatni, míg az RGBI feliratúba a 80 oszlopost. A kettős monitorokat, mint pl. a Commodore 1902-es, amely mind 40, mind 80 oszlopos kijelzésre alkalmas, bármelyik csatlakozóba be lehet dugni.

Monitortípusok

Kompozit monitorok

A kompozit monitorokat 40 oszlopos képernyőkijelzésre tervezték. Ilyenek pl. a Commodore 1701-es és 1702-es monitorok, amelyek minden 40 oszlopos programhoz mindhárom módban használhatók, 80 oszlopos kijelzésre azonban nem.

RGBI monitorok

Ezeket kifejezetten a 80 oszlopos kijelzésre tervezték. Bár az RGBI a *R*ed *G*reen *B*lue *I*ntensity (vörös, zöld, kék, intenzitás) rövidítése, az RGBI monitorok lehetnek színesek vagy egyszínűek. A legnépszerűbb egyszínű monitorok a zöld vagy sárga színt használják. Az RGBI aljzathoz csatlakoztatott RGBI monitor mind C128-as, mind CP/M üzemmódban képes 80 oszlopos kijelzésre.

Kettős (dual) monitorok

A kettős monitorok, mint pl. a Commodore 1902-es, egyaránt képesek összetett video (40 oszlop) és RGBI (80 oszlop) kijelzésre. A kettős monitorokat bármelyik csatlakozóba be lehet dugni. A monitoron található kapcsolóval, ill. a számítógép bekapcsolása után a 40/80 billentyűvel lehet váltani a képernyőkijelzést. Győződjön meg arról, hogy a számítógép 40/80 billentyűjének állása megfelel-e a monitor elején található csúszókapcsoló állásának.
Megjegyzés: A 40/80 billentyű állásától függetlenül is lehet a 40 és a 80 oszlopos kijelzés között váltani; nyomjuk le, majd engedjük fel az ESC billentyűt és nyomjuk meg az X billentyűt.

GYÁRI CSOMAGOLÁSÚ 80 OSZLOPOS SZOFTVER HASZNÁLATA

A legtöbb CP/M program csakúgy, mint a többi C128-as üzemmódban használatos, üzleti felhasználásra kerülő program, 80 oszlopos képer-

nyőt használ. Miután egy normál nyomtatott oldal is 80 oszlop széles, a 80 oszlopos szövegfeldolgozó ugyanúgy jeleníti meg az információt a képernyőn, mint ahogy a papíron meg fog jelenni. A spreadsheet programok is gyakran 80 oszlopos formátumúak, hogy legyen elég hely az oszlopoknak és az információs rubrikáknak. A legtöbb adatbázis programhoz és telekommunikációs programhoz is 80 oszlopos képernyőre van szükség.

80 OSZLOPOS PROGRAMOK KÉSZÍTÉSE

Az előrecsomagolt szoftverek futtatásán kívül a 80 oszlopos képernyőszélesség hasznos lehet saját programok készítéséhez is. Valószínűleg már észrevette, hogy mi történik, ha a 40 oszlopos képernyőn 40 oszlopnál szélesebb sort ír be. A sorok „befordulnak", azaz a következő sorban folytatódnak. Ez a sorok olvasásánál problémát jelenthet, sőt programozásbeli hibákhoz is vezethet, amelyek 80 oszlopos képernyővel kiküszöbölhetők. Általában véve a 80 oszlopos képernyő tisztább kijelzést és jobb áttekintést ad.

40 ÉS 80 OSZLOP EGYÜTTES HASZNÁLATA

A 40 oszlopos összetett videokijelzés legfőbb előnye a bittérképes grafika, míg a 80 oszlopos a szövegfeldolgozáshoz és egyéb üzleti célú felhasználáshoz alkalmasabb. Ha két monitorja van, „osztott" programokat is készíthet, a szöveges részeknél a 80 oszlop előnyeit használhatja ki, míg a grafikánál a 40 oszlopéit. A GRAPHIC 1,1 speciális paranccsal program közben is át lehet váltani a 40 oszlopos képernyőre. Ha kettős monitorja van, amely mind 40, mind 80 oszlop megjelenítésére képes, a programba beiktatott GRAPHIC 1,1 utasítással a grafika 40 oszlopon jeleníthető meg. Ehhez azonban a monitor videokapcsolóját 40 oszlopra kell állítani. Ha ilyen programot ír, jó ötlet, ha a használó számára utasítást iktat be, hogy kapcsolja át a videokapcsolót.

Pl. ha olyan adatokat kell beírnia a használónak a programjába, amelyek egy egyoszlopos grafikonba kerülnek, a "VALTSON AT 40 OSZLOPRA, HOGY LASSA A GRAFIKONT" üzenet közli a használóval, hogy váltson üzemmódot és így láthatja az eredményt.

```
5 REM **** P.14 ****
10 GRAPHIC 5,1:REM EZ A PARANCS ATKAPCSOL 80 OSZLOPOS SZOVEG UZEMMODBA
20 PRINT"▮40 OSZLOPOS OUTPUT KEZDETE":PRINT
30 PRINT"FORDITSA AZ 1902 DUAL MONITOR ELEJEN LEVO KAPCSOLOT KOZEPSO
ALLASBA"
40 PRINT:PRINT"HA EZ MEGTORTENT, NYOMJA MEG A RETURN GOMBOT"
50 GRAPHIC0,1
60 PRINT:PRINT"NYOMJA MEG A RETURN GOMBOT,HA KESZEN VAN":GETKEY A$:
IF A$<>CHR$(1 3)THEN 60
70 COLOR 1,5:COLOR 4,1:COLOR 0,1
80 GRAPHIC 2,1:CHAR 1,2,18,"BIT TERKEP/SZOVEG OSZTOTT KEPERNYO"
:REM OSZTOTT KEPE
RNYO KIVALASZTASA
90 FOR I=70 TO 220 STEP 20 :CIRCLE 1,I,50,30,30:NEXT
100 PRINT"KAPCSOLJON 80 OSZLOPOS OUTPUTRA"
110 PRINT"FORDITSA A KAPCSOLOT A MONITOR ELEJEN A LEGSZELSO ALLASBA"
120 PRINT"NYOMJON RETURN-T HA KESZEN VAN":GETKEY A$:IF A$<> CHR$(13) THEN 120
130 GRAPHIC 5,1: REM EZ AZ UTASITAS 80 OSZLOPOS SZOVEG UZEMMODBA KAPCSOL
140 FOR J=1 TO 10
150 PRINT"ON MOST 80 OSZLOPOS SZOVEG UZEMMODBAN VAN"
160 NEXT:PRINT
170 PRINT"MOST KAPCSOLJON VISSZA 40 OSZLOPOS UZEMMODBA":PRINT
```

```
180 PRINT"KAPCSOLJA A MONITOR ELEJEN LEVO KAPCSOLOT KOZEPSO ALLASBA"
190 PRINT"NYOMJA A RETURN-T, HA KESZEN VAN":GETKEY A$:IF A$<> CHR$(13)
THEN 190
200 GRAPHIC 0,1:REM EZ AZ UTASITAS 40 OSZLOPOS SZOVEG MODBA KAPCSOL"
210 FOR J=1 TO 70
220 PRINT"ON MOST 40 OSZLOPOS UZEMMODBAN VAN"
230 NEXT
```

Mint előbb már említettük, a 40/80 oszlopos formátumok között a gép bekapcsolt állapotában az ESC/X billentyűkkel válthat.
A következő mintaprogram azt mutatja be, hogyan lehet egy programon belül a kettős képernyőt használni.
Mindkét képernyőformátumnak megvannak az előnyei, és a kettőt lehet egy programon belül is alkalmazni, hogy kiegészítsék egymást. 40 oszlopos képernyő használatával kiválóan megjeleníthető a bonyolult BASIC grafika, a 80 oszlopos kijelzés pedig több helyet biztosít a saját programjai számára, ezen kívül futtathatja a 80 oszlopos képernyőre tervezett számos szoftvert is.

....

Ebből a fejezetből megismerhette a Commodore 128-as számítógép C128-as üzemmódjának lehetőségeit. A következő fejezet arról szól, hogyan lehet a számítógépet C64-es üzemmódban használni.

III. FEJEZET

A C64-ES ÜZEMMÓD

9. ALFEJEZET

A billentyűzet használata C64-es üzemmódban

A BASIC 2.0 használata 94

A billentyűzet karakterkészlete 94

Az írógép jellegű billentyűk 94

A *parancsbillentyűk* 94

A kurzor mozgatása C64-es üzemmódban 94

A funkcióbillentyűk programozása C64-es üzemmódban ... 94

A BASIC 2.0. HASZNÁLATA

A Commodore 64-es számítógépbe épített BASIC 2.0 nyelv teljes egészében benne foglaltatik a Commodore 128 BASIC 7.0 nyelvében. A BASIC 2.0 parancsokat mind C128-as, mind pedig C64-es módban használhatjuk (I. a II. fejezet 3. és 4. alfejezetét).

A BILLENTYŰZET KARAKTERKÉSZLETE

A szóköz billentyű fölötti billentyűzetet – a külön álló legfelső sor 3 billentyűcsoportja kivételével – és a jobb oldali billentyűzet felső négy billentyűjét lehet C64-es módban használni. C64-es módban a billentyűzetnek ugyanaz a két karakterkészlete van, mint C128-as módban.

- nagybetű/grafikus karakter;
- nagybetű/kisbetűs karakter.

Ha C64-es módba vált, a billentyűzet nagybetű/grafikus karakterre van beállítva, tehát minden szöveg, amelyet beír, nagybetűkkel jelenik meg. C64-es módban egyszerre csak egy karakterkészletet lehet használni. Ha át akar váltani, a SHIFT és a C= billentyűket kell egyszerre lenyomni.

AZ ÍRÓGÉP JELLEGŰ BILLENTYŰK

Hasonlóan a C128-as módhoz, az írógép jellegű billentyűket C64-es módban is lehet nagy- és kisbetűk írására használni. A főbillentyűzet felső sorában található számbillentyűket, valamint a billentyűk homloklapján található grafikus jeleket is használhatja.

A PARANCSBILLENTYŰK

A legtöbb parancsbillentyű (amelyekkel a számítógépet utasítani lehet, pl. a RETURN, a SHIFT, a CTRL stb.) ugyanúgy működik, mint C128-as üzemmódban. Az egyetlen különbség, hogy C64-es üzemmódban a kurzort csak a főbillentyűzet jobb alsó sarkában található két kurzorbillentyűvel mozgathatja. (C128-as módban a főbillentyűzet felső része fölött elhelyezett négy, nyíllal ellátott billentyűt is használhatja.)

A KURZOR MOZGATÁSA C64-ES ÜZEMMÓDBAN

C64-es üzemmódban a főbillentyűzet két CRSR billentyűjét, valamint a SHIFT billentyűt kell használni a kurzor mozgatására, a 3. alfejezetben leírtak szerint.

A FUNKCIÓBILLENTYŰK PROGRAMOZÁSA C64-ES MÓDBAN

A billentyűzet jobb oldalán, a számbillentyűzet fölött található a négy *funkcióbillentyű*. A tetejükön a felirat F1, F3, F5 és F7, a homloklapjukon pedig F2, F4, F6 és F8. Ezeket a billentyűket programozni lehet, azaz megtanítani valamilyen különleges feladat, funkció elvégzésére, ellátására, ezért *programozható funkcióbillentyűk*nek is nevezik őket.
Ahhoz, hogy a számítógép végrehajtsa a billentyűk homloklapján található jelzéssel – F2, F4, F6, F8 – kapcsolatos feladatokat, a SHIFT billentyűt lenyomva kell tartani. Ezeket tehát *váltott* (SHIFTed) *programozható funkcióbillentyűk*nek nevezzük.
C64-es módban a funkcióbillentyűkhöz nincs hozzárendelve nyomtatott karakter, csak CHR$ kódok. Minden billentyűnek két CHR$ kódja van, az egyik arra az esetre, amikor a billentyűt magában nyomjuk le, a másik pedig, ha a SHIFT billentyűvel együtt használjuk. A páros számú funkcióbillentyűket kell a SHIFT-tel együtt használni, tehát az F2-t úgy kapjuk meg, hogy a SHIFT-tel egyidejűleg nyomjuk meg az F1-et.
Az F1–F8 billentyűk CHR$ kódjai 133-tól 140-ig terjednek, azonban nem numerikus sorrendben. A billentyűknek megfelelő CHR$ kódok a következők:

```
F1 CHR$(133)
F2 CHR$(137)
F3 CHR$(134)
F4 CHR$(138)
F5 CHR$(135)
F6 CHR$(139)
F7 CHR$(136)
F8 CHR$(140)
```

A funkcióbillentyűket többféleképpen használhatjuk a programozásban. Ehhez a 4. alfejezetben leírt GET utasítás ismerete szükséges. Pl. a következő program az F1 billentyűt arra készíti föl, hogy üzenetet írjon ki a képernyőre.

```
10 ?"NYOMD MEG AZ F1-ET, HA FOLYTATNI
AKAROD"
20 GET A$:IF A$="" THEN 20
30 IF A$< >CHR$(133) THEN 20
40 ?"AZ F1-ET NYOMTAD MEG"
```

Ez a program a 20-as és a 30-as sorban végzi el a munka lényegét. A 20-as sor arra szólítja fel a számítógépet, hogy várjon, amíg egy billentyűt le nem nyomunk. Figyelje meg, hogy ha a THEN után közvetlenül egy GOTO parancs következik, akkor elég csak a sorszámot megadni. Azt is jegyezze meg, hogy a GOTO parancs ugyanarra a sorra is vonatkozhat, mint amelyikben szerepel. A 30-as sor azt mondja a számítógépnek, hogy menjen vissza és várjon, amíg egy másik billentyűt le nem nyomunk, abban az esetben, ha nem az F1-et nyomtuk meg.

10. ALFEJEZET

Saját programok tárolása és használata C64-es üzemmódban

Lemez formálása C64-es módban

A SAVE parancs
Kimentés lemezre 96
Kimentés kazettára 96

A LOAD és a RUN parancs
Betöltés és futtatás lemezről 96
Betöltés és futtatás kazettáról 96

Egyéb, lemezzel kapcsolatos parancsok
A program ellenőrzése 97
A lemez tartalomjegyzékének megjelenítése 97
A lemezegység inicializálása 97

Ha egyszer megszerkesztett egy programot, minden bizonnyal tartósan tárolni akarja, hogy később ismét használhassa. Ehhez vagy Commodore lemezegységre, vagy Datasette-re (kazettás egységre) van szüksége.

LEMEZ FORMÁLÁSA C64-ES MÓDBAN

Ha új lemezen akar adatokat tárolni, először elő kell készíteni a lemezt az adatok befogadására. Ezt nevezzük a lemez formálásának. Mindig kapcsolja be előbb a lemezegységet, mielőtt lemezt helyezne bele. Üres lemez formálásához a következő parancsot írja be:

OPEN 15,8,15: PRINT#15, "SZAM:NEV,AZONOSITO" **RETURN**

A név helyére tetszőleges nevet írjon, 16 karakter hosszúságig. Az azonosító helyett bármilyen kétkarakteres kód állhat.
A formálás közben a kurzor eltűnik. Ha ismét villog, írja be a következő parancsot:

CLOSE 15 **RETURN**

Megjegyzés: Ha egy lemezt C64-es vagy C128-as módban formáltunk, bármelyik módban felhasználható.

A SAVE PARANCS

A SAVE paranccsal a programot lemezen vagy szalagon lehet tárolni.

Kimentés lemezre

Ha Commodore szóló lemezegysége van, a következőt kell beírnia, ha lemezen akarja tárolni a programot:

SAVE "PROGRAMNEV",8 **RETURN**

A 8-as azt közli a számítógéppel, hogy lemezegység segítségével tárolunk.
A "PROGRAMNEV"-re ugyanazok a szabályok vonatkoznak akár lemezen, akár szalagon tárolunk. Ez tetszőleges név lehet, betűk, számok és/vagy jelek kombinációja, együttvéve maximum 16 karakter. A kurzor a kimentés alatt eltűnik, és ha a kimentésnek vége van, ismét megjelenik.

Kimentés kazettára

Ha a tároláshoz Datasette-et használ, helyezzen bele egy üres kazettát a magnetofonba, ha szükséges, orsózza vissza a szalagot, és írja be a következőt:

SAVE "PROGRAMNEV" **RETURN**

Miután egy programot kimentett, be lehet tölteni újra a számítógép tárába, és bármikor ismét lehet futtatni.

A LOAD ÉS A RUN PARANCS

Betöltés és futtatás lemezről

Ha lemezről szeretne programot betölteni, írja be:

LOAD "PROGRAMNEV",8 **RETURN**

Ebben az esetben is a 8-as azt jelzi a számítógépnek, hogy lemezegységgel dolgozik. Ha futtatni akarja a programot, írja be a RUN parancsot és nyomja meg a RETURN billentyűt.

Betöltés és futtatás kazettáról

Ha kazettáról akar programot betölteni, írja be:

LOAD "PROGRAMNEV" **RETURN**

Ha nem tudja a program nevét, azt is beírhatja:

LOAD **RETURN**

és a szalagon található legközelebbi program fog betöltődni.
A Datasette-en található számlálóval azonosíthatja a programok kezdési helyét. Ilyen esetben, ha egy programot be akar tölteni, csévélje előre a szalagot 000-ról a program elejére, és írja be:

LOAD **RETURN**

Ilyenkor nem kell megadni a programnevet, hanem az automatikusan töltődik, mivel az lesz a kazettán található legközelebbi program.
Megjegyzés: A töltési folyamat alatt a betöltött program nem törlődik a szalagról, csak átmásolódik a számítógépbe. Egy program betöltése azonban automatikusan töröl minden, a számítógép tárában található BASIC programot.

EGYÉB, LEMEZZEL KAPCSOLATOS PARANCSOK

A program ellenőrzése

Ha azt kívánja ellenőrizni, hogy egy programot helyesen mentett-e ki vagy töltött-e be, írja be a következőt:

VERIFY"PROGRAMNEV", 8 **RETURN**

Ha a számítógépben levő program azonos a lemezen találhatóval, a képernyőn az OK üzenet jelenik meg.
A VERIFY (ellenőrizd) parancs a szalagon tárolt programok esetében is működik. Írja be ezt:

VERIFY"PROGRAMNEV" **RETURN**

Ne feledje, hogy sem a vessző, sem a 8-as szám nem kell, mert a 8-as azt jelenti, hogy lemezzel dolgozik.

A lemez tartalomjegyzékének megjelenítése

Ha látni kívánja a lemezen található programok listáját, írja be:

LOAD "$",8 **RETURN**

A folyamat időtartamára eltűnik a kurzor. Amikor a kurzor megjelenik, írja be:

LIST **RETURN**

Ekkor megjelenik a tartalomjegyzék. *Figyelem!* Ha a tartalomjegyzéket betölti, a tárban levő valamennyi program törlődik.

A lemezegység inicializálása

Ha a lemezegység jelzőfénye villog, ez hibát jelez. Az ún. *inicializálási* folyamattal vissza lehet állítani a lemezegységet a hiba előtti állapotába. Ehhez írja be a következőt:

OPEN 1,8,15, "I":CLOSE 1 **RETURN**

Ha a jelzőfény még mindig villog, vegye ki a lemezt, majd kapcsolja ki és ismét be a lemezegységet.
Ha többet szeretne megtudni a programok kimentéséről és betöltéséről, lapozza fel a lemezegység vagy a Datassette kézikönyvét, valamint az V. fejezetben található BASIC kislexikon LOAD és SAVE címszavait.

IV. FEJEZET

A CP/M ÜZEMMÓD

11. ALFEJEZET

Bevezetés a CP/M 3.0 operációs rendszerbe

Mi a CP/M?

Mit kell tudni a CP/M használatáról?

Első lépések a CP/M 3.0-val

A CP/M 3.0 betöltése 102

A CP/M bejelentkezése a képernyőn 102

A parancssor

A parancsok típusai 103

Hogyan olvassa a CP/M a parancssorokat 103

MI A CP/M?

A CP/M a Digital Research Inc. cég terméke. A Commodore 128-as által használt CP/M változat a CP/M Plus 3.0. Ebben a fejezetben általában CP/M 3.0-ként vagy egyszerűen CP/M-ként említjük. A fejezet általános áttekintést ad arról, hogyan használható a CP/M a Commodore 128-ason.

A CP/M 3.0 a mikroszámítógépek népszerű működési rendszere. Mint ilyen, vezérli és ellenőrzi a számítógép tárát és lemeztárát, konzolját (képernyő és billentyűzet) és távközlési berendezéseit, valamint a lemezfile-okban található információt. A CP/M 3.0 segítségével lemezről file-okat lehet a számítógép tárába átmásolni, vagy olyan perifériákba, mint pl. a nyomtató. A CP/M különféle programokat helyez el a tárba, és a beírt parancsnak megfelelően hajtja végre őket. Ha egyszer már a tárban van egy program, lépések sorozatán keresztül végre lehet hajtani vele bizonyos feladatokat.

A CP/M-et használhatja saját programok készítéséhez vagy választhat a számos CP/M felhasználói program közül.

MIT KELL TUDNI A CP/M HASZNÁLATÁRÓL?

A CP/M 3.0-hoz a következő hardver szükséges: egy Z80-as mikroprocesszort tartalmazó számítógép, egy billentyűzetből, valamint képernyőből álló konzol és legalább egy lemezmeghajtó egység. A CP/M 3.0-hoz szükséges Z80-as mikroprocesszor a C128-asba be van építve; a konzol a C128-as esetében a teljes billentyűzet és egy 80 oszlopos monitor; a szükséges lemezmeghajtó egység pedig az új gyors Commodore 1571-es. Ezen túl két CP/M lemez is a számítógép tartozéka: az egyik a CP/M 3.0 rendszert és egy bővített HELP segédprogramot tartalmaz; a másik pedig számos egyéb segédprogramot.

Megjegyzés: Noha a CP/M használható 80 oszlopos monitorral, egy időben csak 40 oszlop jeleníthető meg a képernyőn. 80 oszlop megtekintéséhez a képernyőt függőlegesen kell görgetni. Ezt a CONTROL billentyű és a megfelelő (bal vagy jobb) kurzorbillentyű lenyomásával végezheti el.

ELSŐ LÉPÉSEK A CP/M 3.0-VAL

A következőkben ismertetjük, hogyan kell betölteni a CP/M 3.0-t, hogyan kell szerkeszteni a parancssorban és hogyan kell tartalék másolatot készíteni a CP/M 3.0 lemezről.

A CP/M 3.0 betöltése

A CP/M 3.0 betöltése azt jelenti, hogy a CP/M 3.0 rendszerlemezen levő operációs rendszert a számítógép beolvassa a tárába.

Többféleképpen lehet a CP/M 3.0-t betölteni. Ha a számítógép ki van kapcsolva, először kapcsolja be a lemezegységet, helyezze bele a CP/M 3.0 rendszerlemezt, majd kapcsolja be a számítógépet. A CP/M 3.0 automatikusan betöltődik. Ha már C128 BASIC üzemmódban van, akkor úgy lehet betölteni a CP/M 3.0-t, hogy a CP/M rendszerlemezt belehelyezi a lemezegységbe és beírja a BOOT BASIC parancsot, vagy úgy, hogy a rendszerlemez behelyezése után megnyomja a RESET gombot.

Ha C64-es módban van, és így akar átváltani CP/M módba, először kapcsolja ki a számítógépet, majd helyezze be a CP/M rendszerlemezt az egységbe és kapcsolja be a számítógépet.

Vigyázat! Mielőtt az ajtaját becsukja, mindig győződjön meg arról, hogy a lemezt teljesen berakta az 1571-es lemezegységbe. CP/M 3.0 módban a Commodore 128-ason a használónak 59 K TPA (*T*ransient *P*rogram *A*rea) áll rendelkezésére, amely valójában felhasználói RAM.

A CP/M bejelentkezése a képernyőn

Miután betöltöttük a CP/M 3.0-t a tárba, az bejelentkezik a képernyőn. Ennek fontos része a következő karakteres üzenet:

A>

Ez a CP/M *rendszer prompt*ja, ez közli a használóval, hogy a CP/M készen áll arra, hogy a billentyűzeten beírt parancsot olvassa, ezenkívül azt is, hogy az A lemezegység az alapegység. Ez azt jelenti, hogy amíg nem mond mást a CP/M-nek, az az A lemezegységben levő lemezen keresi a programokat és az adatfile-okat. Közli továbbá, hogy Ön a 0 jelű használó, mert más használói számot nem találunk.

Megjegyzés: CP/M módban a szóló lemezegység az A meghajtó. Ez azonos a C128-as és a C64-es módban a 8 egységszámmal és a 0 meghajtóegységgel. Általában a CP/M 3.0-nak maximálisan négy meghajtóegysége lehet. A továbbiakat B, C stb. meghajtónak nevezzük.

A PARANCSSOR

A CP/M 3.0 a billentyűzeten beírt különleges parancsok szerint végzi el a feladatokat. Ezek egy ún. parancssorban jelennek meg a képernyőn. A CP/M 3.0 parancssor a *parancs kulcsszó*ból és az opcionális *végződés*ből áll. A kulcsszó az elvégzendő parancs (program) azonosítására szol-

gál. A végződés külön információkat tartalmazhat, pl. a file nevét vagy egyéb paramétereket. A következőkben egy parancssort mutatunk be:

A>*DIR MYFILE*

A használó által beírt karaktereket a fejezetben végig dőlt betűből szedtük, hogy megkülönböztessük a rendszer karakterkijelzéseitől. A példában a DIR a parancs kulcsszava, a MYFILE a végződés. Hogy a CP/M 3.0 parancssora dolgozni kezdjen, nyomja meg a RETURN billentyűt, amelyet a következőképpen jelölünk: **RETURN**
A billentyűzeten beírt karakterek megjelennek a képernyőn. Írás közben a kurzor jobbra mozog. Ha gépelés közben hibázik, vagy az INST/DEL billentyűt, vagy a CTRL-H-t nyomja meg, s így a kurzor balra mozog és a hibát ki lehet javítani. A CTRL a CONTROL billentyű rövidítése. Vezérlőkarakter kijelöléséhez tartsa lenyomva a CTRL billentyűt és nyomja meg a megfelelő betűvel ellátott billentyűt. (A vezérlőkarakterek listája és használatuk a 13. alfejezetben található.)
A kulcsszót és a végződést nagy- és kisbetűk bármilyen kombinációjában beírhatja. A CP/M 3.0 a parancssorban levő minden betűt nagybetűként kezel. A parancssort általában közvetlenül a rendszer promptja után kell beírni, azonban a CP/M 3.0 arra is lehetőséget ad, hogy a prompt és a kulcsszó között üres karakterhelyet hagyjunk.

A parancsok típusai

A CP/M 3.0 kétféle parancsot ismer: beépített és ideiglenes felhasználású parancsot. A beépített parancsok olyan programokat hajtanak végre, amelyek a tárban mint a CP/M működési rendszer része szerepelnek. A beépített parancsokat azonnal végre lehet hajtatni. Az ideiglenes felhasználású parancsokat lemezen, programfileként tároljuk. Előbb be kell őket tölteni a lemezről, hogy feladatukat elvégezhessék. Az ideiglenes felhasználású programfile-okat arról lehet felismerni a tartalomjegyzék megjelenítésekor, hogy a file-név után pont és a COM betűk állnak. (.COM). A 14. alfejezet tartalmazza a CP/M beépített és ideiglenes felhasználású parancsait. Ideiglenes felhasználáskor a CP/M 3.0 csak a kulcsszót ellenőrzi. Sok esetben egyedi parancsvégződésre van szükség. Ha ezt is hozzáírjuk, a CP/M 3.0 ellenőrzés nélkül továbbítja. A végződés maximálisan 128 karaktert tartalmazhat.

Hogyan olvassa a CP/M a parancssorokat?

Használjuk a DIR parancsot annak a bemutatására, hogy a CP/M hogyan olvassa a parancssorokat. A DIR, amely a DIRectory (tartalomjegyzék) rövidítése, azt közli a CP/M-mel, hogy írja ki a képernyőre a lemezfile-ok tartalomjegyzékét. Írja be a DIR kulcsszót a rendszer bejelentkezése után, és nyomja meg a RETURN billentyűt:

A>*DIR* **RETURN**

A CP/M erre a parancsra az A lemezegységben található lemezen levő összes file nevét kiírja. Pl. ha az A meghajtóban a CP/M rendszerlemez van, ilyen katalógus fog megjelenni a képernyőn:

A: PIP COM : ED COM: CCP COM : HELP
COM : HELP HLP
A: DIR COM : CPM SYS

A CP/M kizárólag helyesen írt parancsszavakat ismer föl. Ha gépelési hibát vét, és a kijavítás előtt nyomja meg a RETURN billentyűt, a CP/M 3.0 megismétli, „visszhangozza" a parancssort, és utána kérdőjelet tesz. Pl. tegyük föl, hogy hibásan gépelte a DIR parancsot, mint a következő példában:

A>*DJR* **RETURN**

A CP/M így válaszol:

DJR?

Ez arra utal, hogy a CP/M nem talál DJR kulcsszót. Ha az ilyen gépelési hibákat ki akarja javítani, az INST/DEL billentyűt kell használnia a helytelen betűk törlésére. A karakterek törlésének másik módja, hogy lent tartja a CTRL billentyűt és a H billentyűvel a kurzort balra mozgatja. CP/M-ben egy sor más vezérlőkarakter is van a parancssorok szerkesztésére. A 13. alfejezetből megtudhatja, hogyan kell a vezérlőkaraktereket parancssorok és egyéb információk szerkesztésére használni. A DIR után végződésként a filenév következik. A DIR-t egy file-névvel arra is használhatja, hogy megtudja, rajta van-e egy bizonyos file a lemezen. Pl. ha ellenőrizni akarja, hogy a MYFILE nevű file program a lemezen van-e, írja be:

A>*DIR MYFILE* **RETURN**

A CP/M ezt a feladatot úgy hajtja végre, hogy vagy az illető file nevét írja ki a képernyőre, vagy pedig a következő üzenetet:

NO FILE (nincs file)

Ne feledjen legalább egy üres karakterhelyet beírni a DIR után, hogy a kulcsszót elválassza a végződéstől. Ha ez nem történik meg, a CP/M 3.0 így felel:
A>*DIRMYFILE* **RETURN**
DIRMYFILE?

12. ALFEJEZET

File-ok, lemezek és meghajtók CP/M 3.0 módban

Mi az a file?

A file készítése

Egy file elnevezése
- **A file megjelölése (specifikáció)** 106
- **A használói szám** 107
- **Joker karakterek használata egynél több file kiválasztására** 107
- **A fenntartott karakterek** 107
- **A fenntartott file-típusok** 107

Hogyan lehet CP/M 3.0 lemezekről és file-okról másolatokat készíteni?
- **Másolás szóló lemezegységgel** 108
- **Másolás kettős lemezegységgel** 108

MI AZ A FILE?

A CP/M egyik legfontosabb feladata, hogy lemezein file-okat tegyen hozzáférhetővé és tartson fenn. CP/M módban alapvetően ugyanolyanok a file-ok, mint C128-as vagy C64-es módban, tehát információgyűjtemények. Azonban a CP/M kissé másképpen kezeli a file-okat, mint a C128-as vagy a C64-es módok. Ebben a fejezetben leírjuk a CP/M által használt file-ok két típusát, a file-készítés, -elnevezés és -használat módjait, és azt, hogy hogyan tároljuk a CP/M lemezeken a file-okat.
Mint már megjegyeztük, a CP/M 3.0 file egy információgyűjtemény. Minden file-nak egyedi nevet kell adni, csak így tudja a CP/M azonosítani a file-t. Minden lemez tartalomjegyzéket is tartalmaz, amelyen a lemezen található file-ok neve és helye van felsorolva.
Kétféle CP/M file van, program- (parancs-) file és adatfile. A programfile utasítások sorozatából áll, amelyeket a számítógép lépésről lépésre követ a kívánt eredmény elérése érdekében. Az adatfile járulékos információk gyűjteménye (pl. nevek, címek listája, raktárkészlet, könyvelési adatok, dokumentum szövege stb.).

A FILE KÉSZÍTÉSE

A file készítésének többféle módja van, az egyik a szövegszerkesztő (text editor) használata. A CP/M ED szövegszerkesztője file készítésére és elnevezésére alkalmas. Úgy is lehet file-t készíteni, hogy egy már meglevő file-t új helyre másolunk át, új névvel. CP/M módban a PIP paranccsal lehet átmásolni és újra elnevezni file-okat.
Az ED és PIP parancsokat, valamint a többi általánosan használt CP/M parancsot a 14. alfejezet foglalja össze. Ezek és a többi CP/M parancs részletes ismertetése a CP/M Plus Felhasználói kézikönyvben találhatók meg.

EGY FILE ELNEVEZÉSE

A file megjelölése (specifikáció)

A CP/M minden file-t egyedi megjelölés útján azonosít. Ennek négy része lehet: a lemezegység betűjele, a file-név, a file-típus és a jelző. Az egyetlen kötelező rész a file-név.

A lemezegység betűjele

Egyetlen betű (A–P), amely után kettőspont áll. A rendszer minden lemezegységének más betű a jele. A CP/M-et arra szólítjuk fel, hogy a megjelölt lemezegységben keressen az épp benne levő lemezen egy file-t. Pl. ha beírjuk a következőt:

B:*MYFILE* **RETURN**

akkor a CP/M a B lemezegységben keresi a MYFILE-t. Ha kihagyjuk a lemezegység betűjelét, a CP/M 3.0 az alapegységben, az A-ban keresi az illető file-t.

A file-név

1–8 karakterből állhat, pl.

MYFILE

Előfordulhat, hogy egy file specifikációja mindössze a file-névből áll. Úgy válassza meg a nevet, hogy lehetőleg a file tartalmáról is mondjon valamit. Pl. ha egy üzlethez szükséges programban a vevők nevének listája szerepel a file-on, a neve lehet

VEVŐK

úgy, hogy a file-névből fogalmat alkothassunk annak tartalmáról.

A file-típus

Az ugyanabba a kategóriába tartozó file-ok között segít eligazodni a CP/M opcionális, 1, 2 vagy 3 karakter hosszúságú bővítménye, amelyet a file-névhez lehet illeszteni. Ennek a neve file-típus. Ha a file-névhez file-típust ad hozzá, a kettőt ponttal válassza el egymástól. Próbáljon meg olyan betűket használni, amelyek elárulnak valamit a file-ról, pl. a vevők nevének file-ja a VEVŐ. NÉV file-típust kaphatja.
Amikor a CP/M kiírja a specifikációkat, a rövid file-nevekben üres karakterhelyeket hagy, hogy gyorsan össze lehessen hasonlítani a file-típusokat. Azoknak a programfile-oknak, amelyeket a CP/M lemezről tölt a tárba, COM file-típusa van.

A jelszó

A Commodore 128 CP/M 3.0 rendszerében a file meghatározásába egy jelszót is be lehet iktatni. A jelszó 1–8 karakter hosszú lehet. Ha alkalmazza, válassza el a file-típustól (ennek hiánya esetén a file-névtől) egy pontosvesszővel a következők szerint:

VEVŐ.NÉV; SZÁMLA

A jelszó opcionális, azonban ha egy file-nak jelszava van, akkor mindenképpen be kell írnia a jelszót, ha hozzá akar jutni a file-hoz.

A file-megjelölés mintája

A mind a négy lehetséges elemet tartalmazó file megjelölésében a következők vannak: a lemezegység betűjele, az elsődleges file-név, a file-típus és a jelszó, a következő példán látható karakterek, ill. jelek által elválasztva:

A:DOKUMENTUM.TORVENY; JUDIT **RETURN**

A használói szám

A CP/M tovább azonosítja a file-okat úgy, hogy mindegyiknek ad egy 0-tól 15-ig terjedő használói számot. Ezt akkor rendeli hozzá a file-hoz, amikor a file elkészül. A használói számok lehetővé teszik, hogy 16 file-csoportra osszuk a file-okat. A használói szám mindig megelőzi a lemezegység betűjelét, kivéve a 0 jelű használót, ez az alapszám és ezt a beköszönő prompt üzenet nem írja ki. Íme néhány példa a használói számra és jelentésükre.

4A>	4-es használói szám	A lemezegység
A>	0-s használói szám	A lemezegység
2B>	2-es használói szám	B lemezegység

A beépített USER paranccsal lehet megváltoztatni a mindenkori használói számot:

A> USER 3 **RETURN**
3A>

A legtöbb parancs csak azt a file-t találja meg, amelyiknek az éppen aktuális használói szám van. Azonban, ha a file a 0 használói számon van és rendszerfile tulajdonsága van, bármelyik file-számmal elérhető.

Joker karakterek használata egynél több file kiválasztására

Bizonyos CP/M 3.0 beépített és ideiglenes parancsok több file-t is ki tudnak választani és fel tudnak dolgozni, ha speciális joker karaktereket írunk a file-névbe vagy a file-típusba. A joker olyan karakter, amelyet más karakterek helyett lehet beírni. A CP/M 3.0 a csillagot (*) és a kérdőjelet (?) használja jokernek. Pl. ha ?-et használ egy file-név harmadik karaktereként, akkor ezzel azt mondta a CP/M-nek, hogy a ? helyén bármilyen karakter állhat. Hasonlóképpen, a * azt közli a CP/M-mel, hogy töltse meg a file-nevet kérdőjelekkel. A jokert tartalmazó file-megjelölést határozatlan specifikációnak nevezzük, mivel egynél több file-ra is vonatkozhat. Ezáltal egy mintát adunk a CP/M-nek, s az a tartalomjegyzékben található valamennyi, a mintának megfelelő file-t kiválasztja. Pl. ha beírja a következőt:

?????TAX.LIB

akkor a CP/M 3.0 minden olyan file-t kiválaszt, amelynek a file-neve TAX-ra végződik és file-típusa .LIB.

A fenntartott karakterek

A 12.1. táblázatban szereplő karaktereknek CP/M 3.0-ban speciális jelentésük van, tehát ne használja ezeket a karaktereket másképp, csak a jelzett módon.

12.1. táblázat

A CP/M 3.0 fenntartott karakterei

Karakter	*Jelentés*
<$, !]>[] tab betűhöz kocsivissza	– file-specifikáció elválasztók

12.1. táblázat folytatása

Karakter	*Jelentés*
:	– lemezegység elválasztó file-specifikációnál
.	– file-típus elválasztó file-specifikációnál
;	– jelszó elválasztó file-specifikációnál
;	– megjegyzés elválasztó parancssor elején
* ?	– joker karakter határozatlan file-specifikáció esetén
< > & !]¢	– elválasztók alternatívák (opciók) felsorolásánál
[]	– elválasztók alternatívák felsorolásánál globális és helyi alternatívák esetén
()	– többszörös módosítók elválasztói szögletes zárójelen belül, módosítóval ellátott alternatívák részére
/ $	– alternatívák elválasztói parancssorban

A fenntartott file-típusok

A CP/M 3.0 már létrehozott néhány file-csoportot. A 12.2. táblázat felsorolja a file-típusokat, és rövid ismertetésüket.

12.2. táblázat
A CP/M 3.0 fenntartott file-típusai

File-típus	*Jelentés*
ASM	assembler forrás-file
BAS	BASIC forrásprogram
COM	Z80 vagy ezzel egyenértékű gépi nyelvű program
HEX	a MAC output file-ja (a HEXCOM használatában)
HLP	HELP üzenet-file
$$$	időszakos file
PRN	MAC vagy RMAC nyomtató file
REL	A RMAC output file-ja (a LINK használatában)
SUB	a SUBMIT által végrehajtott parancsok listája
SYM	a MAC, RMAC vagy LINK jelfile-ja
SYS	rendszer-file

HOGYAN LEHET CP/M LEMEZEKRŐL ÉS FILE-OKRÓL MÁSOLATOKAT KÉSZÍTENI?

CP/M 3.0 lemezeiről másolatokat is készíthet egy vagy két lemezegységgel. A másolólemezek lehetnek újak vagy használtak is. Formálhat új lemezeket, vagy pedig használt lemezeket a megfelelő CP/M lemezformáló program segítségével átformálhat. Ha a lemezeket előzőleg már használták, győződjön meg arról, hogy a lemezeken nincsenek más file-ok.
A másolatok készítéséhez használja a CP/M rendszerlemezen található COPYSYS és PIP programokat. A PIP minden programot és adat-file-t le tud másolni, a Copysys csak az operációs rendszer másolására alkalmas.

Másolás szóló lemezegységgel

Szóló Commodore 1541-es vagy 1571-es lemezegységgel is le lehet másolni egy lemez tartalmát a másikra. A PIP parancs az egyik file tartalmát átmásolja a másikra. A PIP formátumának ellenőrzésére írja be a következőt:

A>*HELP PIP*

Válaszul a számítógép megadja PIP parancs szintaxisát. Használja forrásnak az A lemezegységet, célegységnek pedig az E-t. Az E lemezegységet virtuális lemezegységnek nevezzük, azaz nem létezik mint valóságos hardver. A másolási folyamat alatt olyan üzenetet fog kapni, hogy vegye ki a forráslemezt és helyettesítse a céllemezzel.

Másolás kettős lemezegységgel

A következőkből megtudhatja, hogyan kell másolatokat készíteni két meghajtóval (A és B). A-tól P-ig más két betű is lehet a lemezegység betűjele. Ha a CP/M 3.0 rendszerlemezről akar másolatot készíteni, először használja a COPYSYS-t a működési rendszer töltőjének lemásolására.

Győződjön meg arról, hogy a rendszerlemez az A lemezegységben van, amely az alapegység, és az üres lemez pedig a B-ben, majd írja be a következő parancsot:

A>COPYSYS

A CP/M 3.0 betölti a COPYSYS-t a tárba és futtatja. A COPYSYS a következőket jeleníti meg a képernyőn. Ha a program beköszön, csak akkor nyomja meg a RETURN billentyűt, ha ellenőrizte, hogy a megfelelő lemezegységben van.

```
COPYSYS VER 3.0
SOURCE DRIVE NAME (OR RETURN FOR
DEFAULT) ?A
SOURCE ON A THEN TYPE RETURN
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO
REBOOT) ?B
DESTINATION ON B THEN TYPE RETURN
FUNCITON COMPLETE
DO YOU WISH TO COPY CPM.SYS ? YES
```

(azaz:

- Copysys 3.0 változat;
- a forrásegység neve (vagy vissza az alapegységhez) ?A;
- ha a forrás az A, írja be a RETURN-t;
- művelet befejezve;
- célegység neve (vagy vissza az újratöltéshez) ?B;
- ha a cél a B, írja be a RETURN-t;
- művelet befejezve;
- le akarja másolni a CPM.SYS-t? Igen).

(A CP/M a fenti üzeneteket ismétli a CP/MSYS másoláskor.)

A>

Most csak az operációs rendszerről van másolata. Ha a rendszerlemez többi file-ját is le akarja másolni, írja be a következő PIP parancsot:

A>*PIP B:=A:*,**

Ez a PIP parancs az összes, a tartalomjegyzékben található file-t az A lemezegységről a B-re másolja. A PIP a COPYING (másolok) üzenetet írja ki minden file-név után, a másolási folyamat alatt. Amikor a PIP befejezte a másolást, a CP/M 3.0 kiírja a rendszer prompt üzenetét.

Most a B lemezegységen van a rendszerlemezről egy pontos másolata. Vegye ki az eredeti rendszerlemezt az A egységből és helyezze biztonságba. Amíg az eredetit változatlan állapotban megőrzi, helyre tudja állítani a CP/M programfile-okat, ha a munkakópiával történne valami.

13. ALFEJEZET

A konzol és a nyomtató használata CP/M üzemmódban

A konzol outputjának (kivitelének) vezérlése 112

A nyomtató kivitelének vezérlése 112

Sorszerkesztés a konzolon 112

Vezérlőkarakterek használata sorszerkesztésnél 112

A következő részből azt tudhatja meg, hogyan kommunikál a CP/M a konzollal és a nyomtatóval, hogyan kell beindítani és megállítani a konzol és a nyomtató kivitelét és hogyan lehet parancsokat szerkeszteni.

A KONZOL OUTPUTJÁNAK (KIVITELÉNEK) VEZÉRLÉSE

Előfordul, hogy a CP/M 3.0 túl gyorsan jeleníti meg az információt a képernyőn, úgy, hogy a használónak nincs ideje elolvasni. Ha meg akarja kérni a rendszert, hogy várjon, amíg Ön a kiírást elolvassa, tartsa lenyomva a CTRL billentyűt és nyomja meg az S-et. E billentyűk egymás utáni használata szünetet iktat be a kiírásban. Ha készen van, nyomja meg a CTRL-Q-t, hogy a kiírás folytatódjon. A NO SCROLL billentyű lenyomása is szünteltеti a rendszert és egy *szünet* ablakot helyez a képernyő alján levő status sorba (25. sor). Ha a kiírást folytatni akarja, nyomja meg ismét a NO SCROLL billentyűt. Ha a CTRL-Q vagy a NO SCROLL billentyűkön kívül bármi mást is lenyom a kiírás szünetében, a CP/M megszólaltatja a konzol csengőjét.
Néhány CP/M segédprogram (mint pl. a DIR és a TYPE) automatikusan követi az oldalakat a konzolon. Ez azt jelenti, hogy ha a kiírt program hosszabb, mint amennyi egyszerre ráfér a képernyőre, a kiírás automatikusan megáll, ha a képernyő tele van. Ha ez bekövetkezik, a CP/M olyan üzenetet ad, hogy nyomja meg a RETURN-t, amennyiben folytatni akarja a kiírást. Ezt a lehetőséget a SETDEF paranccsal használhatja, ill. ki is iktathatja.

A NYOMTATÓ KIVITELÉNEK VEZÉRLÉSE

Arra is van mód, hogy egy paranccsal vezérelje, hogy a konzol kiírása a nyomtatón is jelenjen meg *(visszhangozzék)*. Ehhez nyomja meg a CTRL-P-t. Egy bip hang közli, hogy bekapcsolt a nyomtató. Ha meg akarja állítani, nyomja meg ismét a CTRL-P-t. (Ekkor már nincs bip hangjelzés.) Mialatt a nyomtató működik, bármilyen karakter, amely megjelenik a képernyőn, szerepelni fog a nyomtatón is.
A nyomtatót a DIR paranccsal együtt a lemezen tárolt file-ok felsorolására is használhatja. A CTRL-P-t a CTRL-S-sel és a CTRL-Q-val használva a file egy részéről is készíthet papírmásolatot. A TYPE paranccsal a konzolon beindíthatja a file kijelzését. Amikor a kinyomtatni kívánt részhez ér, nyomja meg a CTRL-S-et, ekkor a kiírás megáll, majd a CTRL-P-t, mire a nyomtató működni kezd. Ezután nyomja meg a CTRL-Q-t, hogy a kiírás folytatódjon és beinduljon a nyomtatás. A végén egy újabb CTRL-S, STRL-P és CTRL-Q sorozattal befejezheti a nyomtatást.

SORSZERKESZTÉS A KONZOLON

Mint ahogy már említettük, az egyszerű gépelési hibákat az INST/DEL vagy a CTRL-H billentyűvel lehet kijavítani. A CP/M-nek további sorszerkesztő funkciói vannak, amelyeket a vezérlőkarakterekkel lehet végrehajtani. A vezérlőkaraktereket használjuk a legtöbb program parancssorainak vagy adatbeviteli sorainak szerkesztéséhez.

VEZÉRLŐKARAKTEREK HASZNÁLATA SORSZERKESZTÉSNÉL

A 13.1. táblázatban felsorolt sorszerkesztésre alkalmas vezérlőkarakterek használatával a kurzort balra és jobbra lehet mozgatni, ezáltal a parancssorba karaktereket tudunk beszúrni, ill. onnan kitörölni. Így a javított helytől jobbra nem kell mindent újra írni.
A következő mintapéldában a használó rosszul írta a PIP-et és a CP/M 3.0 hibaüzenettel válaszolt. A használó visszahívja a hibás parancssort a CTRL–W billentyűkkel, és kijavítja a hibát.

A>POP A:=B:*.*	(a PIP rosszul van írva)
POP?	
A>POP A:=B:*.*	(a CTRL–W visszahívja a sort)
A>POP A:=B:*.*	(a CTRL–B a sor elejére viszi a kurzort)
A>POP A:=B:*.*	(a CTRL–F jobbra viszi a kurzort)
A>PP A:=B:*.*	(a CTRL–G törli a hibát)
A>PIP A:=B:*.*	(a beírt I betű kijavítja a parancssort)

Miután kijavította a parancssort, megnyomhatja a RETURN billentyűt akkor is, ha a kurzor a sor közepén van. A RETURN billentyű lenyomása (vagy a megfelelő vezérlőkarakterek egyikéé) nemcsak hogy végrehajtja a parancsot, de tárolja is a pufferban, úgyhogy vissza lehet hívni szerkesztésre, vagy újra végre lehet hajtatni a CTRL–W billentyű lenyomásával.
Ha egy sor közepére karaktert szúr be, a kurzortól jobbra levő karakterek jobbra tolódnak. Ha a sor hosszabb lesz, mint a képernyő szélessége, a karakterek a képernyő jobb szélén eltűnnek. Ezek a karakterek azonban nem vesznek el, hanem újra megjelennek, ha a sorból karaktereket törlünk vagy ha megnyomjuk a CTRL–E-t, amikor a kurzor a sor közepén van. A CTRL–E minden, a kurzortól jobbra levő karaktert a képernyő következő sorába visz.

A 13.1. táblázat a Commodore 128 CP/M 3.0 rendszerében a sorszerkesztő vezérlőkarakterek teljes listáját közli.

13.1. táblázat

CP/M 3.0 sorszerkesztő vezérlőkarakterek

Karakter	*Jelentés*
CTRL–A	A kurzort egy karakterrel balra viszi.
CTRL–B	A kurzort a parancssor elejére viszi anélkül, hogy a sor tartalmán változtatna. Ha a kurzor a sor elején van, a CTRL–B a sor végére viszi.
CTRL–E	Fizikai kocsivissza, de nem küldi a parancssort CP/M 3.0-ba. A kurzort a következő sor elejére viszi anélkül, hogy az előző bevitelt törölné.
CTRL–F	A kurzort egy karakterhellyel jobbra viszi.
CTRL–G	A kurzor alatti karaktert törli. A kurzor nem mozdul el, a tőle jobbra levő karakterek eggyel balra kerülnek.
CTRL–H	A kurzortól balra levő karaktereket törli és egy karakterhellyel balra viszi a kurzort. Az attól jobbra levő karakterek egy hellyel balra kerülnek.
CTRL–I	A kurzort a következő tabulátorállásba viszi. Ezek automatikusan minden nyolcadik oszlopnál vannak. A CTRL–I-nek ugyanaz a hatása, mint a TAB billentyűnek.
CTRL–J	A parancssort CP/M 3.0-ba küldi, a kurzort pedig az új sor elejére. Ugyanúgy működik, mint a RETURN vagy a CTRL–M billentyű.

13.1. táblázat folytatása

Karakter	*Jelentés*
CTRL–K	A kurzortól a sor végéig töröl.
CTRL–M	A parancssort CP/M-be küldi, a kurzort pedig az új sor elejére. Ugyanúgy működik, mint a RETURN vagy a CTRL–J billentyű.
CTRL–R	Újra megjeleníti a parancssort. A kurzor pillanatnyi helyére egy karaktert helyez, és minden eddig beírt részparancsot újraír.
CTRL–U	A parancssorból félretesz minden karaktert, egy #-et tesz a kurzor helyére, és a kurzort a következő sorba viszi. A CTRL–W-vel azonban minden karaktert vissza lehet hívni, amely a CTRL–U lenyomásakor a kurzortól balra állt.
CTRL–W	A korábban beírt parancssort visszahívja és kiírja mind az operációs rendszer szintjén, mind pedig a programok végrehajtása közben, ha a CTRL–W az első beírt karakter a prompt üzenet megjelenése után. A CTRL–J, a CTRL–M, a CTRL–U és a RETURN definiálják a parancssort, amelyet vissza lehet hívni. Ha a parancssor karaktereket tartalmaz, a CTRL–W a kurzort a parancssor végére viszi. Ha megnyomjuk a RETURN-t, a CP/M 3.0 végrehajtja a parancsot.
CTRL–X	A kurzortól balra levő összes karaktert félreteszi és a kurzort az aktuális sor elejére viszi. A CTRL–X a kurzortól jobbra minden karaktert megőriz.

14. ALFEJEZET

A legfontosabb CP/M 3.0 parancsok összefoglalása

A CP/M 3.0 parancsok két fajtája 116

Beépített parancsok 116

Időszakos felhasználású parancsok 116

A bevitel és a kivitel átirányítása 117

Logikai eszközök hozzárendelése 117

Programfile-ok megkeresése 117

Többszörös parancsok végrehajtása 118

Programok befejezése 118

Segítségkérés 118

Mint ahogyan a 11. alfejezetben már említettük, a CP/M 3.0 parancssora parancskulcsszóból, opcionális végződésből és a RETURN billentyű lenyomásából áll. Ebben az alfejezetben arról a kétfajta parancsról lesz szó, amelyeket a kulcsszó azonosítani képes, valamint összefoglaljuk az egyedi parancsokat és funkcióikat. Ezen kívül példákon keresztül bemutatunk néhány általánosan használt parancsot. Ugyancsak ez a rész tartalmazza a CP/M 3.0 logikai és fizikai eszközeinek felsorolását is, valamint azt, hogy miként keresi meg a CP/M 3.0 a lemezen a programfile-t, hogyan lehet többszörös parancsokat végrehajtani és a lemezrendszert visszaállítani kiindulási helyzetébe. Végezetül ez az alfejezet magyarázza meg azt is, hogyan kell a HELP (segítség) parancsot használni, hogy különféle CP/M témákban – mint pl. a parancsok formátuma és használatuk – a billentyűzet mellett ülve is tájékozódhassunk.

A CP/M 3.0 PARANCSOK KÉT FAJTÁJA

CP/M 3.0 módban kétféle parancs van:

- *beépített parancsok* (ezek a tárban azonosítanak programokat);
- *nem beépített,* ún. időszakos felhasználású parancsok (ezek lemezen azonosítanak programfile-okat).

A CP/M 3.0-nak hat beépített és több mint húsz időszakos felhasználású parancsa van. További időszakos felhasználású parancsokat lehet a rendszerhez hozzáadni különféle CP/M 3.0 kompatíbilis használói programok beszerzésével.

Ha már gyakorlott programozó, saját programokat is írhat, amelyek CP/M 3.0 módban működnek.

A BEÉPÍTETT PARANCSOK

Ezek a CP/M 3.0 mindig használható elemei, függetlenül attól, hogy melyik lemezegységben milyen lemez van. A beépített parancsok akkor kerülnek a számítógép tárába, amikor a CP/M betöltődik. Így ezek a programok gyorsabban teljesítik feladatukat, mint az időszakosok. A 14.1. táblázat a Commodore 128-as számítógép CP/M 3.0 beépített parancsait sorolja fel.

Néhány beépített parancsnak vannak opcionális részei is, amelyeket egy hozzájuk kapcsolódó átmeneti programmal lehet megadni. A kapcsolt időszakos parancsnak azonos neve kell hogy legyen, mint a beépített parancsnak, file-típusa pedig mindig COM.

14.1. táblázat
Beépített parancsok

Parancs	*Funkció*
DIR	A tartalomjegyzék valamennyi file-nevét kiírja, kivéve a SYS-szel jelzetteket.
DIRSYS	A SYS-szel (system = rendszer) jelzett file-ok nevét írja ki a tartalomjegyzékből.
ERASE	Kitörli a file nevét a tartalomjegyzékből és felszabadítja a file által elfoglalt tárhelyet.
RENAME	Átnevezi a lemez egy file-ját.
TYPE	Az ASCII (TEXT = szöveg) -file tartalmát kiírja a képernyőre.
USER	Más használói számra vált.

IDŐSZAKOS FELHASZNÁLÁSÚ PARANCSOK

A CP/M 3.0 időszakos felhasználású parancsait a 14.2. táblázat tartalmazza. Ha beírjuk az időszakos parancsot azonosító kulcsszót, a CP/M 3.0 betölti a lemezről a programfile-t, és a végződésbe beírt file-nevet, adatokat vagy paramétereket hozzárendeli a file-hoz.

14.2. táblázat
Időszakos felhasználású parancsok

Név	*Funkció*
COPYSYS	Új betöltőlemezt készít.
DATE	Beállítja vagy kiírja a dátumot és az időt.
DEVICE	Logikai CP/M eszközöket rendel hozzá egy vagy több fizikai eszközhöz, vált az eszközmeghajtó protokoll és baud (bit/s) arányain, valamint beállítja a konzol képernyőméretét.
DIR	Kiírja a tartalomjegyzéket a file-okkal és jellemzőikkel.
DUMP	ASCII és hexadecimális formában írja ki a file-okat.
ED	ASCII file-okat készít és változtat meg.
ERASE	Joker törlésére használható.
GET	Ideiglenesen a konzolbevitelt a lemezfile-ról fogadja el a billentyűzet helyett.
HELP	Kiírja a CP/M parancsokról szóló információt.
INTDIR	A tartalomjegyzéket inicializálja az idő és dátum kijelzés lehetővé tételéhez.

PIP	File-okat másol és fűz össze.
PUT	Ideiglenesen a lemezfile-ba irányítja a nyomtató vagy a konzol kivitelét.
RENAME	Joker karakterek felhasználásával megváltoztatja egy file vagy file-csoport nevét.
SET	Beállítja a file-opciókat: lemezcímkék, file-tulajdonságok, az idő és a dátum kijelzés típusa, jelszó védelem.
SETDEF	Beállítja a rendszer-opciókat.
SHOW	Lemez és meghajtó statisztikát ír ki.
SUBMIT	Automatikusan végrehajt többszörös parancsokat.
TYPE	Szövegfile (vagy jokerkarakterek használata esetén filecsoportok) tartalmát írja ki a képernyőre, vagy ha szükséges, kinyomtatja.

A BEVITEL ÉS KIVITEL ÁTIRÁNYÍTÁSA

A CP/M 3.0 PUT parancsával a konzol vagy a nyomtató kiírását lemezfile-ba lehet irányítani. A GET paranccsal a CP/M 3.0-ba vagy egy időszakos programba konzol bevitelt lehet lemezfile-ból végrehajtani. A következő példák a GET és a PUT néhány lehetőségét világítják meg.
A PUT paranccsal a konzol kivitelét lemezfile-ra és konzolra is lehet irányítani. A PUT-tal lemezfile-t lehet készíteni, amelyben a lemezen található valamennyi file katalógusa megtalálható, mint az a 14.1. ábrán látható.

```
A>PUT CONSOLE OUTPUT TO FILE DIR.PRN
PUTTING CONSOLE OUTPUT TO FILE: DIR. PRN

A>DIR
A:FILE-  TEX:     TEX:      BAK:ONE  BAK:     TEX
NAME     FRONT    FRONT              THREE
A:FOUR   TEX:ONE  TEX:LINE- TEX:     TXT:TWO  BAK
                  DIT       EXAMP1
A:TWO    TEX:     BAK:      TXT
         THREE    EXAMP2

A>TYPE DIR.PRN
A:FILE-  TEX:     TEX:      BAK:ONE  BAK:     TEX
NAME     FRONT    FRONT              THREE
A:FOUR   TEX:ONE  TEX:LINE- TEX:     TXT:TWO  BAK
                  DIT       EXAMP1
A:TWO    TEX:     BAK:      TXT
         THREE    EXAMP2
```

14.1. ábra. Péda a PUT parancsra

A GET paranccsal a CP/M 3.0-t vagy egy programot úgy lehet irányítani, hogy a billentyűzet helyett lemezfile-ról olvasson be konzol bevitelt. Ha a file-t a CP/M 3.0 olvassa, szabvány CP/M 3.0 parancssorokat kell tartalmaznia. Ha a file-t időszakos program olvassa, akkor a programnak megfelelő bevitelt kell tartalmaznia. Egy file tartalmazhat mind CP/M 3.0 programsorokat, mind pedig programbemenetet, amennyiben a program elindítására vonatkozó parancsot is tartalmaz.

LOGIKAI ESZKÖZÖK HOZZÁRENDELÉSE

A Commodore 128 CP/M 3.0-hoz szükséges minimális hardver a következő: billentyűzetből és képernyőből álló konzol, valamint egy 1571-es lemezegység. Előfordulhat, hogy egyéb eszközt kíván a géphez kapcsolni, pl. nyomtatót vagy modemet. Hogy ezeket a fizikailag különböző beviteli és kiviteli eszközöket nyomon lehessen követni, a 14.3. táblázat megadja a CP/M 3.0 logikai eszközeinek elnevezését, valamint a Commodore 128-as rendszerben az ezekhez rendelt fizikai eszközöket.

14.3. táblázat

A CP/M 3.0 logikai eszközei

A logikai eszköz neve	Az eszköz típusa	A hozzárendelt fizikai eszköz neve
CONIN	Konzol bevitel	Billentyűzet
CONOUT	Konzol kivitel	80 oszlopos képernyő
AUXIN	Kisegítő bevitel	–
AUXOUT	Kisegítő kivitel	–
LST	Lista kivitel	PTR1 vagy PTR2

A hozzárendeléseket a DEVICE paranccsal lehet megváltoztatni. Pl. az AUXIN-t és az AUXOUT-ot modemhez lehet rendelni, hogy a számítógép telefonvonalakon keresztül más használókkal, a Compuserve információs szolgálattal vagy komputerizált hírszolgálattal kapcsolatba kerülhessen.

PROGRAMFILE-OK MEGKERESÉSE

Ha egy parancs kulcsszó időszakos felhasználású programot azonosít, a CP/M 3.0 az illető programfile-t az alapegységben vagy a specifikált lemezegységben keresi. Az aktuális használói szám alatt, majd a 0 használói szám alatt keresi ugyanazt a file-t, SYS kiterjesztéssel. A keresési

folyamat bármely pontján a CP/M 3.0 abbahagyja a keresést, ha a programfile-t megtalálta. Ekkor a CP/M betölti a programot a tárba és végrehajtja. Ha a program lefutott, a CP/M 3.0 kiírja a rendszer prompt üzenetét és várja a következő parancsot. Azonban ha a CP/M 3.0 nem találja a parancsfile-t, akkor megismétli a parancssort egy kérdőjellel a végén, és várja a következő parancsot.

TÖBBSZÖRÖS PARANCSOK VÉGREHAJTÁSA

Az edigi példákban a CP/M 3.0 egyidejűleg csak egy parancsot hajtott végre. A CP/M 3.0 képes azonban parancsok sorozatának végrehajtására is. A rendszer prompt üzenetének megjelenésekor be lehet írni egy sor parancsot is, vagy lemezfile-ba is bevihetünk egy gyakran használt parancssort SUB file-típus (kiterjesztés) használatával. Ha egyszer egy sorozatot lemezfile-ban tárolt, a SUBMIT paranccsal bármikor, ha szüksége van rá, végre lehet hajtani a sorozatot.

PROGRAMOK BEFEJEZÉSE

A CTRL–C kétbillentyűs paranccsal lehet befejezni a program végrehajtását vagy alaphelyzetbe visszaállítani a lemezrendszert. A CTRL–C parancs beviteléhez nyomja le a CTRL billentyűt, majd a C billentyűt. A legtöbb, CP/M alatt futó alkalmazási programot és időszakos felhasználású programot be lehet fejezni a CTRL–C billentyűk lenyomásával, azonban ha akkor akar programot befejezni, miközben az a képernyőre épp kiír, lehet, hogy a CTRL–C előtt meg kell nyomnia a CTRL–S-et.

SEGÍTSÉGKÉRÉS

A CP/M 3.0-nak van egy időszakos felhasználású parancsa, a HELP, amely a legáltalánosabb CP/M parancsok formátumának és használatának összefoglalását írja ki. Ehhez egyszerűen írja be a következő parancsot:

A>HELP

vagy nyomja meg a HELP billentyűt a HELP szó beírása és a RETURN billentyű megnyomása helyett. Az így elérhető témakörök a következőképpen jelennek meg:

COM-MANDS	CNTRLCH-ARS	COPYSYS	DATE	DEVICE	DIR
DUMP	ED	ERASE	FILESPEC	GENCOM	GET
HELP	HEXCOM	INITDIR	LIB	LINK	MAC
PATCH	PIP (COPY)	PUT	RENAME	RMAC	SAVE
SET	SETDEF	SHOW	SID	SUBMIT	TYPE
USER	XREF				

Tegyük föl, hogy azt írja be:

HELP>PIP

A CP/M ekkor a következő felvilágosítással szolgál a képernyőn:

```
PIP (COPY)
SYNTAX:
     DESTINATION              SOURCE
PIP d: {Gn} | filespec {[Gn]} = filespec
{[O]},... | d:{[O]}
```

Magyarázat: A file-másoló PIP program file-okat másol, fűz össze és közvetít lemezek, nyomtatók, konzolok vagy más, a géphez kapcsolt eszközök között. Az első file-specifikáció a CEL (destination). A második a FORRÁS (source). Ha két vagy több file-t akar összefűzni, használjon egymástól vesszőkkel elválasztott két vagy több forrásfile-specifikációt. [O] az elérhető opcionális lehetőségek (alternatívák) tetszőleges kombinációja A [Gn] alternatíva a célfile-specifikációban azt közli a PIP-pel, hogy arra a felhasználói területre másolja a file-t.
*Ha a PIP után nem áll parancs végződés, megjelenik a * üzenet és a gép vár a parancsok sorozatára, amelyeket soronként kell beírni és úgy is hajtja végre a gép. A forrás és a cél lehet bármilyen CP/M. 3.0 logikai eszköz.*
A HELP valamennyi CP/M 3.0 beépített és ideiglenes felhasználású parancsról az előbbiekhez hasonló információval szolgál. Ha egy speciális területről kíván felvilágosítást kapni, írja be a prompt üzenet megjelenése után a HELP tárgyát, pl.:

A>HELP PIP
A>HELP DIRSYS

Bármikor fordulhat a HELP-hez, ha egy bizonyos paranccsal kapcsolatban információhoz akar jutni. De érdemes, csak átnézni a HELP listákat, elmélyíteni CP/M 3.0 tudását.

15. ALFEJEZET

A Commodore 128-as további szolgáltatásai a CP/M 3.0-hoz

A billentyűzet kibővített lehetőségei

Egy billentyű definiálása (jelentésének meghatározása) 120

Füzér definiálása 120

Az ALT mód használata 120

A képernyő kibővített lehetőségei

A Commodore a CP/M 3.0-hoz egy sor további szolgáltatást nyújt. Ezek a kibővítések a CP/M 3.0-hoz igazítják a Commodore 128-as lehetőségeit. A következőkben ezekről a kibővített lehetőségekről lesz szó.

A BILLENTYŰZET KIBŐVÍTETT LEHETŐSÉGEI

Bármelyik billentyűhöz meghatározhatunk egy kódot vagy funkciót, *kivéve* a következő billentyűket:

Bal SHIFT,
Jobb SHIFT,
C=,
RESTORE,
40–80,
CAPS LOCK.

A billentyűzet definícióként ismeri föl a következő speciális funkciókat. Tartsuk lenyomva a CONTROL billentyűt és a jobb SHIFT billentyűt és ezzel egy időben nyomjuk meg a kívánt funkcióbillentyűt.

Billentyű	*Funkció*
BAL KURZORBILLENTYŰ	Definiálja a billentyűt
JOBB KURZORBILLENTYŰ	Füzért definiál
ALT BILLENTYŰ	Bekapcsolja a billentyűszűrőt

Egy billentyű definiálása (jelentésének meghatározása)

A használó definiálhatja a billentyű használata által létrejött kódot. Minden billentyűnek négy lehetséges definíciója van: Normál, Alpha Shift, Shift és Control. Az Alpha Shiftet a C= billentyűvel lehet ki- és bekapcsolni. Ha ebbe a módba váltunk, a képernyő alján egy kis doboz jelenik meg. Az első lenyomott billentyű lesz az, amelyet definiálni fogunk. Megjelenik az ehhez a billentyűhöz rendelt jelenlegi hexadecimális érték, s a használó most beírhatja a billentyű új HEX kódját vagy törölheti egy nem HEX billentyű lenyomásával. A billentyűhöz rendelhető kódok definíciói a következők (ALT módban a kódok visszakerülnek az alkalmazásba, l. az LAT módról írottakat):

Kód	*Funkció*
00h	Zéró (mintha nem is nyomtunk volna meg billentyűt)
01 – 7Fh	Normál ASCII kódok
80h – 9Fh	Hozzárendelt füzér
A0h – AFh	80 oszlopos karakterszín
B0h – BFh	80 oszlopos háttérszín
C0h – CFh	40 oszlopos karakterszín
D0h – DFh	40 oszlopos háttérszín
E0h – EFh	40 oszlopos keretszín
F0h	Be-, ill. kikapcsolja a lemezállapotot
F1h	Rendszer szünet
F2h	(nincs meghatározva)
F3h	40 oszlopos képernyő ablak/jobb
F4h	40 oszlopos képernyő ablak/bal
F5h – FFh	(nincs meghatározva)

Füzér definiálása

Ez a funkció arra ad lehetőséget, hogy egyetlen billentyűhöz egynél több billentyűkódot rendeljünk. Ebben a módban bármely lenyomott billentyű füzérbe helyezkedik. A használó a beírás eredményét a képernyő alján megjelenő hosszú ablakban tekintheti meg.

Megjegyzés: Előfordulhat, hogy néhány billentyű nem jeleníti meg tartalmát. Hogy a használó ellenőrizhesse az adatbevitel folyamatát, a következő öt billentyűfunkció áll rendelkezésére. Nyomja le a CONTROL és a jobb SHIFT billentyűt és a kívánt funkcióbillentyűt.

Billentyű	*Funkció*
RETURN	Teljes füzér definiálása
+ (a főbillentyűzeten)	Üres karakterhelyet szúr be a füzérbe
– (a főbillentyűzeten)	Törli a kurzorkaraktert
Bal nyíl	Kurzor balra
Jobb nyíl	Kurzor jobbra

Az ALT mód használata

Az ALT mód kapcsolófunkciót tölt be (azaz vált a ki- és bekapcsolt helyzet közt). Az alapérték a kikapcsolt helyzet. Ezzel a funkcióval 8 bites kódokat lehet alkalmazásba küldeni.

A KÉPERNYŐ KIBŐVÍTETT LEHETŐSÉGEI

A CP/M 3.0 alapképernyője egy ADM31-es terminálnak felel meg. A következő képernyőfunkciók azonosak az ADM 3A működéssel, amely az ADM31 egyik fajtája.

CTRL G	Csengő
CTRL H	Kurzor balra
CTRL J	Kurzor le

CTRL K	Kurzor föl
CTRL L	Kurzor jobbra
CTRL M	A kurzort a jelenlegi sor elejére viszi (CR)
CTRL Z	A kurzort HOME helyzetbe viszi és törli a képernyőt
ESC = RC	Kurzorhelyzet, ahol R a sorhely (üres karakterhelytől 8-ig értékkel) és a C az oszlophely (üres karakterhelytől 0 értékkel) az állapotsorra vonatkoztatva

További funkciók ADM31 módban:

ESC T, ESC t	Töröl a sor végéig
ESC Y, ESC y	Töröl a képernyő végéig
ESC :, ESC *	A kurzort HOME helyzetbe viszi és törli a képernyőt az állapotsorral együtt
ESC Q	Karakter beszúrása
ESC W	Karakter törlése
ESC E	Sor beszúrása
ESC R	Sor törlése

Megjegyzés: Az ESC ESC ESC színsorszám a 16 szín valamelyikéből kiválasztja a képernyőszint (a színek felsorolását l. a 6.2. ábrán). A színek sorszáma a következő:

20h – 2Fh	Karakterszín
30h – 3Fh	Háttérszín
40h – 4Fh	Keretszín (csak 40 oszlopnál)

A következő funkciókhoz kapcsolódó vizuális hatások csak a 80 oszlopos képernyőformátum esetén láthatók:

ESC>	Fél fényerősség
ESC<	Teljes fényerősség
ESC G4	Inverz bekapcsolása
*ESC G3	Soraláhúzás bekapcsolása
ESC G2	Villogás bekapcsolása
*ESC G1	Karakterkészlet váltó
ESC G0	Kikapcsolja az összes ESC G attribútumot.

**Megjegyzés:* Ez nem normál ADM31-es kombináció.

****

Ebben a fejezetben a CP/M 3.0 szerkezetének és széles körű alkalmazási lehetőségeinek rövid ismertetését adtuk. Amennyiben bővebb felvilágosítást óhajt, tanulmányozza a Digital Research Inc. CP/M Plus Felhasználói kézikönyvét.

V. FEJEZET

BASIC 7.0 KISLEXIKON

16. ALFEJEZET

A kislexikon szerkezete 126

A parancsok és utasitások formátuma 126

A KISLEXIKON SZERKEZETE

Ez a fejezet a BASIC 7.0 nyelvi elemeit sorolja fel. Közli a Commodore 128 BASIC 7.0 nyelv szabályainak (szintaxisának) teljes listáját és magyarázatokat is ad.
A BASIC 7.0 a BASIC 2.0 valamennyi elemét magában foglalja. A BASIC 7.0 új parancsait, utasításait, függvényeit és operátorait félkövér szedéssel közöljük. A BASIC műveletek különféle típusait külön tárgyaljuk, a következők szerint:

1. Parancsok és utasítások
   – programok szerkesztéséhez, tárolásához és törléséhez szükséges parancsok, valamint
   – BASIC programutasítások egy program megszámozott soraiban.
2. Függvények
   – füzér, numerikus, nyomtató függvények.
3. Változók és operátorok
   – változók különböző típusai, legális változónevek, matematikai operátorok és logikai operátorok.
4. Foglalt szavak és jelek
   – A BASIC 7.0 nyelv foglalt szavai és jelei, amelyek más célra nem használhatók.

A PARANCSOK ÉS UTASÍTÁSOK FORMÁTUMA

A kislexikonban a parancsokat és az utasításokat a következő forma szerint közöljük:

Parancs-név	AUTO
Rövid meghatározás	Be-, ill. kikapcsolja az automata sorszámozót
A parancs formátuma	AUTO [sorszám]
A formátum és a használat leírása	Ez a parancs az automata sorszámozót indítja be, ill. kapcsolja ki, megkönnyítve ezáltal a programozást, mert a használó rögtön számozott sorokkal tud dolgozni. Miután az egyes programsorokat a RETURN megnyomásával bevittük, a következő sorszám megjelenik a képernyőn és a kurzor két karakterhelynyi távolságra lesz a sorszámtól. A sorszám argumentuma a sorszámok között kívánt növekményre vonatkozik. Az AUTO parancs argumentum nélkül kikapcsolja az automatikus sorszámozót és futtatja a programot. Ez az utasítás csak direkt módban, programon kívül használható.
Példák:	
AUTO 10	Automatikusan 10-esével megszámozza a programsorokat
AUTO 50	Automatikusan 50-esével megszámozza a programsorokat
AUTO	Kikapcsolja az automatikus sorszámozást

A formátumot megadó sor a következő elemekből áll:

DLOAD	"programnév"	[DO, U8]
kulcsszó	argumentum	további argumentumok (általában opcionálisak)

A parancs vagy utasítás azon részei, amelyeket pontosan a megadott formában kell beírni, nagybetűvel vannak szedve. A használó által megadandó szavak, pl. a "programnév", kisbetűsek. Az idézőjeleket ("") általában a programnév vagy a file-név előtt és után a formátum minta szerinti helyre kell tenni.

A KULCSSZAVAK lefoglalt szavak, ezeket nagybetűvel közli a kislexikon. Lehet teljes alakjukban vagy az elfogadott rövidítéssel használni. (A K) függelékben valamennyi rövidítés megtalálható.) A kulcsszót pontosan kell beírni, mert a pontatlan beírás hibát eredményez. A BASIC és a DOS hibaüzeneteit az A) és a B) függelék tartalmazza.

A kulcsszavak a BASIC nyelv alkotóelemei. Egy parancs vagy utasítás központi részét képezik, közlik a számítógéppel, hogy mit kell tennie. Ezeket változók nevére nem lehet felhasználni. A 20. alfejezetben található a kulcsszavak és jelek felsorolása.

Az ARGUMENTUMOK, más néven paraméterek kisbetűvel vannak szedve. Az argumentumok a kulcsszavak kiegészítői, speciális jelentést adnak a parancsnak vagy utasításnak. Pl. a LOAD kulcsszó azt közli a számítógéppel, hogy töltsön be egy programot, míg az argumentum azt mondja meg, hogy melyik legyen az a program. A második argumentum azt szabja meg, hogy melyik meghajtóról töltse be a gép a programot. Az argumentum lehet file-név, változó, sorszám stb.

SZÖGLETES ZÁRÓJELben ([]) vannak az opcionális argumentumok, paraméterek, amelyek közül a használó szükség szerint választhat.

A CSÚCSOS ZÁRÓJEL (< >) azt jelzi, hogy a használónak a felsorolt argumentumok közül egyet választania *kell*.

A FÜGGŐLEGES VONAL (|) a felsorolt argumentumok egyes tételeit választja el egymástól, amikor a választék csak a felsorolt argumentumokból áll. Ha a függőleges vonal egy szögletes zárójelbe zárt felsorolásban szerepel, mindössze a felsoroltakból lehet választani, de arra is lehetőség van, hogy egyetlen argumentumot se használjunk.

A HÁROM PONT (...) olyan opcionális paramétert, ill. argumentumot jelöl, amelyet egynél többször is meg lehet ismételni.

Az IDÉZŐJEL karakterfüzéreket, file-neveket és más kifejezéseket zár közbe. Ha az argumentumok vannak idézőjelben, az idézőjeleket bele kell venni a parancsba vagy utasításba. Az idézőjel nem a formátum leírásának szokásos eszköze, hanem a parancs vagy utasítás elengedhetetlen része.

ZÁRÓJEL. Ha az argumentum kerek zárójelben van, a zárójelet is bele kell venni a parancsba vagy utasításba, ui. nem a formátum leírásának eszköze, hanem a parancs vagy utasítás elengedhetetlen része.

A VÁLTOZÓ valamilyen érvényes BASIC változónévre vonatkozik, mint pl. X, A$, T% stb.

A KIFEJEZÉS (képlet) valamely érvényes BASIC kifejezést jelöl, pl. A+B+2,.5*(X+3) stb.

APPEND

APPEND# logikai file-szám "file-név"[,Dmeghajtó száma] [<ON,>Ueszköz-szám]

Ez a parancs megnyitja a megadott file-nevű file-t, a mutatót pedig a file végére helyezi. Az ezt követő PRINT# utasításokkal az adatokat függelékként ennek a logikai file-számnak a végére lehet illeszteni. A meghajtó és az eszköz alapértelmezésbeli számai 0, ill. 8. A file-névként használt változókat vagy kifejezéseket zárójelbe kell tenni.

Példák:

APPEND#8, "MYFILE"	Megnyitja a Myfile nevű logikai file-t, hogy az utána következő PRINT# utasításokkal függeléket lehessen hozzá illeszteni.
APPEND#7, (A$), D0,U9	Az A$ változóval jelzett nevű logikai file-t nyitja meg a 0-ás meghajtón, 9-es eszközszámmal, és előkészíti APPEND-re.

AUTO

Be-, ill. kikapcsolja az automata sorszámozót

AUTO [sorszám]

Ez a parancs az automata sorszámozót indítja be, ill. kapcsolja ki, megkönnyítve ezáltal a programozást, mert a használó rögtön számozott sorokkal tud dolgozni. Miután az egyes programsorokat a RETURN megnyomásával bevittük, megjelenik a következő sorszám a képernyőn és a kurzor két karakterhelynyi távolságra lesz a sorszámtól. A sorszám argumentuma a sorszámok kívánt növekményére vonatkozik. A sorszám argumentuma a sorszámok kívánt növekményére vonatkozik. Az AUTO parancs argumentum nélkül kikapcsolja az automatikus sorszámozót és futtatja a programot. Ez az utasítás csak direkt módban, programon kívül használható.

Példák:

AUTO 10	10-esével automatikusan megszámozza a programsorokat.
AUTO 50	50-esével automatikusan megszámozza a programsorokat.
AUTO	Kikapcsolja az automatikus sorszámozást.

BACKUP

Kettős (iker) lemezegységen az egyik lemez teljes tartalmát átmásolja a másikra.

BACKUP forrás Dmeghajtó száma TO cél Dmeghajtó száma [<ON,>Ueszköz-szám]

Ezzel a paranccsal kettős lemezegység használatával a forráslemezről minden file-t le lehet másolni a céllemezre. A BACKUP paranccsal előzetes megformálás nélkül is lehet új lemezt használni. Ez azért lehetséges, mert a BACKUP parancs a lemez valamennyi információját lemásolja, a formátumot is beleértve, azonban a BACKUP parancs minden, már a lemezen levő információt megsemmisít. Ezért, amikor egy már előzőleg használt lemezre készít másolatot, győződjön meg arról, hogy nincs-e rajta olyan program, amelyet meg akar őrizni. Elővigyázatosságból a számítógép megkérdezi, hogy "ARE YOU SURE?" (Biztos?), mielőtt a műveletet elkezdi. Ha másolni akar, nyomja meg az Y billentyűt, ha nem, akkor bármelyik másikat. Ajánlatos minden lemezről biztonsági tartalékmásolatot késziteni arra az esetre, ha az eredeti elveszne vagy megsérülne. L. még a COPY parancsot. Az eszközszám alapértéke 8.

Megjegyzés: Ez a parancs csak kettős lemezmeghajtó egység használata esetén működik.

Példák:

BACKUP D0 TO D1 8-as egységszámú kettős lemezmeghajtóban a 0-s számú meghajtóban levő lemezről az 1-es számúba levőre másol minden file-t.
BACKUP D0 TO D1 ON U9 A 9-es egységszámú lemezmeghajtóban a 0-s számú meghajtóban levő valamennyi file-t az 1-es számúban levőre másolja.

BANK A 0-tól 15-ig számozott 16 bank valamelyikét kiválasztja

BANK bankszám

Ez az utasítás a bankszámot és a Commodore 128-as tárának megfelelő tárkonfigurációt specifikálja. Az alapérték 15. A következőkben bemutatjuk a Commodore 128-as tárának elérhető BANK konfigurációit.

BANK	KONFIGURÁCIÓ
0	Csak RAM(0)
1	Csak RAM (1)
2	Csak RAM(2)
3	Csak RAM(3)
4	Belső (internál) ROM, RAM(0), I/O
5	Belső ROM, RAM(1), I/O
6	Belső ROM, RAM(2), I/O
7	Belső ROM, RAM(3), I/O
8	Külső (externál) ROM, RAM(0), I/O
9	Külső ROM, RAM(1), I/O
10	Külső ROM, RAM(2), I/O
11	Külső ROM, RAM(3), I/O
12	Kernál és belső ROM (ALACSONY), RAM(0), I/O
13	Kernál és külső ROM (ALACSONY), RAM(0), I/O
14	Kernál és BASIC ROM, RAM(0), karakter ROM
15	Kernál és BASIC ROM, RAM(0), I/O

Egy bizonyos bankot úgy lehet megtalálni a gépi nyelvű monitorban pl. hogy beírja a következőt: BANK n, ahol n = 0–15, és monitorra vált. A monitoron belülről a látott tartomány négyjegyű hexadecimális száma elé tegyen még egy hexadecimális számot (0–F). A folyamat részletes leírását a Commodore 128-as programozói kézikönyve tartalmazza.

BEGIN/BEND Feltételes utasítás, hasonló, mint az IF...THEN:ELSE. Úgy van megalkotva, hogy a szerkezet kezdete (BEGIN) és a vége (BEND) közé több programsort is beiktathat. Formátuma a következő:

```
IF feltétel THEN BEGIN: utasítás
utasítás
utasítás BEND: ELSE BEGIN
utasítás
utasítás BEND
```

Például:

```
10 IF X=1 THEN BEGIN:PRINT "X=1 igaz"
20 PRINT "Az utasításnak ez a része akkor van végrehajtva"
30 PRINT "ha X=1"
40 BEND:PRINT "X nem egyenlő eggyel":PRINT "a BEGIN és a BEND közötti
        utasításokat átugorja a gép"
60 PRINT "a program többi része"
```

Ha a 10-es sorban a feltételes (IF...THEN) utasítás igaz, a BEGIN és a BEND kulcsszavak közötti utasításokat hajtja végre a gép, a BEND-del egy sorban levő összes utasítást is beleértve. Ha a 10-es sorban az IF...THEN feltételes utasítás hamis, a program átugorja a BEGIN és a BEND között álló valamennyi utasítást, beleértve a BEND-del azonos programsorban levőket is. A program a BEND-et tartalmazó sor utáni sorban folytatódik. A BEGIN/BEND lényegében a 10–40-es sorokat egyetlen hosszú sorként értelmezi.
Ugyanezek a szabályok érvényesek az ELSE:BEGIN esetben is. Ha feltétel igaz, az ELSE:BEGIN és a BEND között levő valamennyi utasítást végrehajtja a gép, beleértve a BEND-del egy sorban állókat is. Ha a feltétel hamis, a program a BEND-et tartalmazó sor után közvetlenül álló sorban folytatódik.

BLOAD

Egy megadott tárhelytől kezdődően bináris file-t tölt be.

BLOAD "file-név"[,Dmeghajtó száma] [,Ueszközszám] [,Bbankszám] [,Pstartcím]

ahol
- a file-név a használó által megadott név,
- a bankszám a 16 szám valamelyike,
- a startcím az a tárhely, ahol a betöltés kezdődik.

A bináris file olyan program- vagy adatfile, amelyet vagy gépi nyelv monitorral vagy a BSAVE paranccsal kimentettünk. A BLOAD parancs a bináris file-t a startcím által megadott helyre tölti be.

Példák:

BLOAD "SZELLEMEK", B0, P3584 Betölti a "SZELLEMEK" nevű bináris file-t 0 bankszámmal a 3584-es helytől kezdődően.

BLOAD "1ADATOK", D0, U8,B1, P4096 Az "1ADATOK" nevű bináris file-t a 4096-os helyre tölti (a bank 0), a 0-s meghajtóról és a 8-as egységről.

BOOT

Bináris file-ként kimentett programot tölt be és hajt végre.

BOOT "filenév" [,Dmeghajtó száma] [<ON,> Ueszközszám]

A parancs egy bináris file-t tölt be és a meghatározott startcímen megkezdi a végrehajtást. Az eszközszám alapértéke 8, a meghajtóé 0.

Példa:

BOOT	Végrehajtandó programot tölt be (pl. CP/M Plus-t).
BOOT "GRAFIKA 1", D0, U9	Betölti a "GRAFIKA 1" programot a 0-s meghajtóról, a 9-es egységről és végrehajtja.

BOX

Dobozt rajzol a képernyő meghatározott helyére

BOX [színforrás], X1,Y1[,X2, Y2] [,szög] [,kifestés]

ahol:

A színforrás	0 háttérszín 1 előtérszín 2 sokszínű 1 3 sokszínű 2
x1, y1	A bal felső sarok koordinátái
x2, y2	A jobb alsó sarok koordinátái Az x1, y1-gyel szemben; alaphelyzet a PC (pontkurzor) helye
Szög	Az óramutató járásával megegyező fokonkénti forgás; alaphelyzet a 0 fok

Kifestés	Az idomot színesre festi 0 – ne fess 1 – fess (alaphelyzet a 0)

Ezzel az utasítással a használó tetszőleges méretű négyszöget rajzolhat a képernyőre. Az elforgatás a négyszög középpontja körül történik. A pixel (pont) kurzor a BOX utasítás végrehajtása után az x2, y2 ponton van. A színforrás száma 0 vagy 1 standard bittérkép esetén, vagy 2, ill. 3. többszínű bittérkép esetén. L. még a GRAPHIC parancsot a BOX színforrás-számának megfelelő grafikus mód kiválasztásához.
Az x és y értékek a pontkurzort helyezhetik abszolút koordinátákba (pl. 100, 100) vagy az előző helyzethez viszonyított koordinátákba (+20, −10). Az is lehetséges, hogy az egyik tengely koordinátája relatív, míg a másiké abszolút. Az x és y koordináták lehetséges kombinációi a következők:

x,y	abszolút x, abszolút y
+/− x, y	relatív x, abszolút y
x, +/− y	abszolút x, relatív y
+/−x, +/−y	relatív x, relatív y

A pontkurzor bővebb magyarázatáért l. még a LOCATE parancsot.

Példák:

BOX 1, +10, +10	Dobozt rajzol a pontkurzor jelenlegi helyétől 10 ponttal jobbra és 10-zel lejjebb.
BOX 1, 10, 10, 60, 60	Megrajzolja a négyszög oldalait.
BOX, 10, 10, 60, 60, 45, 1	Kifestett, elforgatott dobozt (rombuszt) rajzol.
BOX, 30, 90, 45, 1	Kifestett, elforgatott sokszöget rajzol.

Bármelyik paraméter elhagyható, de a helyére vesszőt kell tenni, mint a két utolsó példa esetében.

Megjegyzés: Ha a szög nagyobb, mint a 360°, az idom lefordul.

BSAVE Bináris file-t ment ki a meghatározott tárhelyekről

BSAVE "file-név" [,Dmeghajtó száma] [,Ueszközszám] [,Bbankszám], Pstartcím TO Pvégcím

ahol:
- a file-név a használó által megadott név;
- a meghajtó száma 0 vagy 1, kettős meghajtóegységen (szóló lemezegység esetében az alapérték 0);
- az eszközszám a lemezegység száma (az alapérték 8);
- a bankszám a használó által megadott szám (0–15);
- a startcím az a kezdőcím, ahonnan a programot kimentettük;
- a végcím az utolsó kimentett cím a tárban.

Ez a parancs ugyanaz, mint a gépi nyelvi monitornál a SAVE parancs.

Példák:

BSAVE "SPRITE ADATOK", BO, P3584, TO P4096	A "SPRITE ADATOK" nevű bináris file-t menti ki a 3584-es és a 4096-os tárhelyek közé (bank 0)
BSAVE "PROGRAM. SCR", DO, U9,BO, P3182, TO P8000	A "PROGRAM.SCR" nevű bináris file-t menti ki a 3182-es és a 8000-es közötti tártartományba (bank 0) a 0-s meghajtón, 9-es egységen.

CATALOG Megjeleníti a lemez tartalomjegyzékét

CATALOG[Dmeghajtó száma][<ON, > Ueszközszám] [,joker füzér]

A CATALOG parancs a megjelölt meghajtóban megjeleníti a lemez tartalomjegyzékét éppúgy, mint a DIRECTORY parancs (l. ott).
Példa:

CATALOG a 0-s meghajtóban megjeleníti a lemez tartalomjegyzékét.

CHAR Karaktereket ír ki a képernyő megjelölt helyére.

CHAR [színforrás], x,y[, füzér][,RVS]

Ez a parancs eredetileg karaktereknek bittérképes képernyőn való megjelenítésére szolgál, de szövegképernyőn is használható. A paraméterek jelentése:

Színforrás	0 – háttér 1 – előtér
x	Karakteroszlop (0–79), 40 oszlopos módban befordul a következő sorba.
y	Karaktersor (0–24).
Füzér	A kiírandó füzér.
RVS (reverse)	Inverz mező kapcsoló (0 ki, 1 be).

A CHAR utasítással bármely képernyő megadott helyén meg lehet jeleníteni szöveget (alfanumerikus füzért). A karakteradatokat a számítógép a karakter ROM területéről olvassa be. A használó adja meg a kezdőhely x és y koordinátáit, valamint a megjeleníteni kívánt szövegfüzért. A színforrás és az inverz forma opcionális.
40 oszlopos formátumban a füzér a következő sorban folytatódik: amennyiben hosszabb, mint 40 oszlop. TEXT (szöveg) módban a CHAR paranccsal kiírt szöveg ugyanúgy működik, mint a PRINT füzér, beleértve a kurzort és a színvezérlést is. Ezek a vezérlőfunkciók a füzéren belül nem működnek, mikor CHAR paranccsal bittérképes módban akarunk szöveget kiírni. A kis- és nagybetűs vezérlések (CHR$ (14) vagy CHR$ (142)) bittérképes módban szintén működnek.

A többszínű karaktereket másképpen kell kezelni, mint a standard karaktereket. Először a COLOR paranccsal határozza meg a többszínű 1-et és többszínű 2-t. Állítsa a grafikus módot többszínűre. A többszínű 1-nél az előtérnek adjon 0 értéket, az inverz mezőnek szintén 0-t. Többszínű 2-nél az előtér színforrás értéke legyen 0, az inverz mező pedig 1. A következő példa vörös háttéren írja ki az előtér karakterszínét. Váltsa át az inverz mezőt 1-re és a karakterek többszínű 2-ben jelennek meg (kék).

```
10 COLOR 2,3:REM többszínű 1 = vörös
20 COLOR 3,7:REM többszínű 2 = kék
30 GRAPHIC 3,1
30 CHAR 0, 10, 10 "SZÖVEG", 0
```

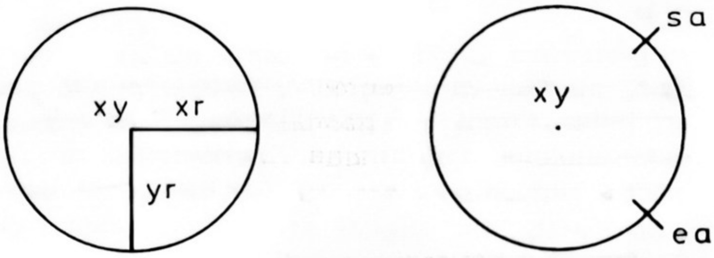
CIRCLE Köröket, ellipsziseket, íveket stb. rajzol a képernyő meghatározott helyére

CIRCLE[színforrás], X,Y[, Xr][, Yr][, sa][, ea][, szög][, inc]

ahol:

a színforrás	0 – háttérszín 1 – előtérszín 2 – többszínű 1 3 – többszínű 2

x, y	A kör középpontjának koordinátái
xr	x sugár
yr	y sugár (az alapérték az x sugár)
sa	Kezdő ívszög (az alapérték 0°)
ea	Végső ívszög (az alapérték 360°)
Angle (szög)	Az óramutató járásának megfelelő forgás szerinti fokok (az alapérték 0°)
inc	A szeletek között fokok (az alapérték 2°)



A CIRCLE utasítással a használó kört, ellipszist, ívet, háromszöget, nyolcszöget vagy egyéb sokszöget rajzolhat. A pontkurzor (PC) a kör kerületén marad, a végső ívszögnél. Minden forgás a középponthoz viszonyul. Ha az y sugarat az x sugárral egyenlővé tesszük, nem kapunk tökéletes kört, mert az x és y koordináták különbözőképpen vannak beállítva (x = 0–320 és y = 0–200). Az íveket a kezdő szögtől az óramutató járásának megfelelően a végső szögig rajzolja a számítógép. A növekmény a forma egyenletességét vezérli – az alacsonyabb értékek a körhöz közelebb álló alakzatokat eredményeznek. Amennyiben a növekménynek (inc) 2-nél nagyobb értéket adunk, durva szélű, szögletes alakzatot kapunk.
Az x és y értékek a pontkurzort helyezhetik abszolút koordinátákba, pl. (100, 100) vagy az előző helyzethez viszonyított koordinátákba, pl. (+20, −10). Az egyik tengely koordinátája lehet relatív, míg a másiké abszolút. Az x és y koordináták lehetséges kombinációi a következők:

x,y	abszolút x, relatív y
+/– x, y	relatív x, abszolút y
x, +/– y	abszolút x, relatív y
+/– x, +/– y	relatív x, relatív y

A pontkurzorról szóló további felvilágosításért l. a LOCATE parancsot.

Példák:

CIRCLE 1, 160, 100, 65, 10	Ellipszist rajzol
CIRCLE 1, 160, 100, 65, 50	Kört rajzol
CIRCLE 1, 60, 40, 20, 18,,,, 45	Nyolcszöget rajzol
CIRCLE 1, 260, 40, 20,,,,, 90	Rombuszt rajzol
CIRCLE 1, 60, 140, 20, 18,,,, 120	Háromszöget rajzol
CIRCLE 1, +2, +2, 50, 50	Kört rajzol, két ponttal lejjebb és kettővel jobbra, mint a pontkurzor eredeti helyzete

Bármelyik paraméter elhagyható, de a vesszőt akkor is ki kell tenni a megfelelő helyre. A meg nem adott paraméterek az alapértéket veszik föl.

CLOSE Zárja a logikai file-t

CLOSE fileszám

Ez az utasítás a DOPEN vagy OPEN utasításokkal megnyitott valamennyi file-t zárja. A CLOSE szót követő szám a bezárandó file száma.

Példa:

CLOSE 2 zárja a 2-es számú logikai file-t

CLR Törli a program változóit.

CLR

Ez az utasítás valamennyi változót töröl a tárban, a programot magát azonban érintetlenül hagyja. A számítógép a RUN vagy a NEW parancsokra az utasítást automatikusan végrehajtja. Szerkesztés után már nem kell a CLR-t használni, mert a változó és a szöveg már nincsenek ugyanabban a tárban.

CMD Átirányítja a képernyőkivitelt.

CMD logikai fileszám [,kiírt lista]

Ez a parancs a kivitelt, amely egyébként a képernyőre menne (a PRINT és a LIST utasításokat, de a POKE-ot nem) más eszközre küldi, pl. a lemez adatfile-ba vagy a nyomtatóhoz. Ezt az eszközt vagy file-t előbb meg kell nyitni (OPEN). A CMD parancs után a file-ra vonatkozó számnak vagy numerikus változónak kell állnia. A kiírt lista bármilyen alfanumerikus füzér vagy változó lehet. Ez a parancs akkor hasznos, ha a programlisták élére fejlécet kell nyomtatni.

Példa:

OPEN 1,4	Megnyitja a 4-es számú eszközt, a nyomtatót.
CMD 1	Minden normál kivitelt a nyomtatóra küld.
LIST	A listázás a nyomtatóra megy, nem a képernyőre, még a READY üzenet is.
PRINT# 1	A kivitelt visszaküldi a képernyőre.
CLOSE 1	Zárja a file-t.

COLLECT Felszabadítja a hozzáférhetetlen lemezhelyet.

COLLECT[Dmeghajtó száma][<ON,> Ueszközszám]

Ezt a parancsot akkor kell használni, ha a lemezen olyan helyet akarunk elérni, amelyet helytelenül zárt (splat) file-hoz foglaltunk le, valamint akkor, ha a tartalomjegyzékből ki akarjuk törölni az ezekre a file-okra vonatkozó referenciákat. A splat-file-ok olyan file-ok, amelyek mellett a tartalomjegyzékben csillag jelenik meg. Az eszközszám alapértéke 8.
Példa:

COLLECT D0 Minden helytelenül zárt file-hoz lefoglalt helyet felszabadít.

COLLISION A sprite ütközések okozta megszakítás kezelésére.

COLLISION típus[, utasítás]

ahol:
A típus a megszakítás típusa:
1 – sprite–sprite ütközés:
2 – sprite–display adat ütközés;
3 – fényceruza (csak 40 oszlop).

Utasítás – a szubrutin BASIC sorszáma.

Ha a megadott helyzet bekövetkezik, a BASIC befejezi az éppen végrehajtott utasítást és egy GOSUB-bal a megadott sorszámra megy. Ha a szubrutinnak vége (RETURN-nel kell befejeződnie), a BASIC ott folytatja a végrehajtást, ahol abbahagyta. A megszakítás addig van érvényben, amíg ugyanolyan típusú, de sorszám nélküli COLLISION utasítást nem adunk meg. A megszakításnak egynél több típusát alkalmazhatjuk egy időben, de kezelni csak egyetlen megszakítást lehet egy időben (azaz nem lehet beágyazott vagy visszatérő megszakításokat alkalmazni). Egy megszakítás egy bizonyos ideig előidézhet további megszakításokat is, ha a helyzet nem változik meg vagy a megszakítást érvényen kívül nem helyezzük.
Ha egy sprite láthatatlan, azaz nincs a képernyőn, nem tud megszakítást generálni. Ha meg akarja határozni, hogy melyik sprite-ok ütköztek az utolsó ellenőrzés óta, használja a BUMP funkciót.

Példa:

COLLISION 0, 5000	Sprite-sprite ütközést, és az 5000-es sorban a szubrutinba küldött vezérlést követi nyomon.
COLLISION 0	A fenti példában megkezdett megszakítást megállítja.
COLLISION 1, 1000	Sprite-adat ütközést, valamint a 1000-es sorban a szubrutinba küldött programvezérlést követi nyomon.

COLOR

Színeket határoz meg a képernyő valamennyi területére

COLOR forrásszám, szín szám.

Ez az utasítás a hét színterület valamelyikéhez rendel hozzá színt.

Terület	*Forrás*
0	40 oszlopos (VIC) háttér
1	40 oszlopos (VIC) előtér
2	Többszínű 1
3	Többszínű 2
4	40 oszlopos (VIC) keret
5	Karakterszín (40 vagy 80 oszlopos)
6	80 oszlopos háttérszín

A használható színek tartománya 1–16.

Színkód	*Szín*	*Színkód*	*Szín*
1	fekete	9	narancs
2	fehér	10	barna
3	vörös	11	világosvörös
4	türkiz	12	sötétszürke
5	bíbor	13	középszürke
6	zöld	14	világoszöld
7	kék	15	világoskék
8	sárga	16	világosszürke

Színszámok 40 oszlopos képernyőformátumnál

Színkód	*Szín*	*Színkód*	*Szín*
1	fekete	9	sötétbíbor
2	fehér	10	sötétsárga
3	sötétvörös	11	világosvörös
4	világostürkiz	12	sötéttürkiz
5	világosbíbor	13	középszürke
6	sötétzöld	14	világoszöld
7	sötétkék	15	világoskék
8	világossárga	16	világosszürke

Színszámok 80 oszlopos képernyőformátumnál
Példa:

COLOR 0,1	A 40 oszlopos képernyő háttérszínét feketére változtatja.
COLOR 5,8	A karakter színét sárgára váltja.

CONCAT Összefűz (láncol) két adatfile-t.

CONCAT "2.file" [, Dmeghajtó száma] TO"1. Fiel" [,Dmeghajtó száma] <ON,> Ueszközszám]

A CONCAT parancs a 2-es számú file-t az 1-es számú végére illeszti és meghagyja az 1. file-nevet. Az eszközszám alapértéke 8, a meghajtóé 0.
Példa:

CONCAT "B file" TO "A file"	A B file-t az A-file-hoz fűzi, és az összekapcsolt file neve *A file* lesz.
CONCAT (A$) TO (B$), D1,U9	A B$ nevű file új file lesz, azonos névvel úgy, hogy az A$ nevű file a B$ végéhez kapcsolódik a 9-es egységen az 1-es meghajtón (kettős lemezmeghajtó).

Ha a file-név egy változó, mint az előző példánál, a file-név változót zárójelbe kell tenni.

CONT Folytatja a program végrehajtását.

CONT

Ezt a parancsot egy, a STOP billentyűvel, a STOP vagy az END utasítással megállított program újrakezdésére használjuk. A program ott folytatódik, ahol megállítottuk. A CONT utasítás nem folytatja a programot, ha a sorokat megváltoztattuk vagy új sorokat adtunk a programhoz vagy ha a képernyőn programszerkesztés történt. Amennyiben a program hiba következtében állt le, vagy ha az újrakezdés előtt követtünk el hibát, a CONT nem működik. Ebben az esetben megjelenik a következő hibaüzenet:
CAN'T CONTINUE ERROR (Nem tudom folytatni, hiba).

COPY Kettős vagy szóló lemezegységen az egyik meghajtóról a másikra másol file-okat.

COPY "forrás file-név" [, Dmeghajtó száma] TO "célfile-név" [, Dmeghajtó száma][<ON,> Ueszközszám]

Ez a parancs az egyik lemezről (forrás-file) a másikra (cél-file) másol file-okat kettős lemezmeghajtó egységen. Szóló egységen ugyanarra a lemezre is képes másolni, de a file-névnek ekkor másnak kell lennie, míg két meghajtó esetében lehet azonos is.
A COPY paranccsal az egyik meghajtóról a másikra le lehet másolni valamenynyi file-t is. Ebben az esetben a meghajtó számát kell megadni és a forrás-, valamint a cél file-neveket el is lehet hagyni. A COPY parancs alapértelmezésbeli paraméterei: 8 – eszközszám; 0 – meghajtó.

Megjegyzés: két szóló vagy két kettős lemezegység között nem lehet másolni, l. a BACKUP parancsot.

Példák:

COPY D0, "teszt" TO D1, "teszt program"	Lemásolja a 0-s meghajtóról az 1-esre a „teszt"-et és az 1-es meghajtón „teszt program" nevet ad ki.
COPY D0, "HOLMI" TO D1, "HOLMI"	A „HOLMI"-t a 0-s meghajtóról az 1-es meghajtóra másolja át.
COPY D0 TO D1	A 0-s meghajtóról az összes file-t átmásolja az 1-es meghajtóra.
COPY "MUNKAPROGRAM" TO "TARTALÉK"	A „MUNKAPROGRAM" nevű file-t „TARTALÉK" néven ugyanarra a lemezre másolja át (0-s meghajtó).

DATA

Meghatározza a program által használt adatokat.

DATA állandók listája

Ezt a parancsot egy adatsor követi, amelyet a READ utasításokkal kell a számítógép tárába beolvasni. Az adatok lehetnek numerikusak vagy füzérek, közöttük vesszők állnak. A füzéradatoknak nem kell idézőjelben állniuk, hacsak nem tartalmaznak üres karakterhelyet, vesszőt vagy kettőspontot. Ha két vessző között nem áll semmi sem, az értéket a számítógép numerikus adat esetén nullának, füzér esetén üres füzérnek olvassa. L. még a RESTORE utasítást, amellyel a Commodore 128-as újra be tud olvasni adatokat.

Példa:

DATA 100, 200, PISTI, "SZIA, ANYU",,3, 14, ABC 123

DCLEAR

A lemezmeghajtón minden nyitott csatornát töröl.

DCLEAD [Dmeghajtó száma][<ON, > Ueszközszám]

Ez az utasítás a megadott eszközszámon minden file-t zár és minden nyitott csatornát töröl. Az alapértékek D0,U8. Egyenértékű az OPEN 10,8,15,"I0" CLOSE 10 paranccsal.

Példák:

DCLEAR D0	A 0-s meghajtón (8-as eszköz) minden nyitott file-t töröl.
DCLEAR D1, U9	Az 1-es meghajtón (9-es eszközszám) minden nyitott file-t (csatornát) töröl.

DCLOSE

Zárja a lemezfile-t.

DCLOSE [#logikai fileszám][<ON, > Ueszközszám]

Ez az utasítás a lemezegységen éppen nyitott egyetlen vagy valamennyi file-t zárja. Ha nem adjuk meg a logikai file-számot, minden éppen nyitott file-t zár. Az eszközszám alapértéke 8.

Példák:

DCLOSE	Zárja a 8-as egységen éppen nyitott file-okat
DCLOSE 2	A 8-as egységen a logikai file-számot viselő file-t zárja
DCLOSE ON U9	A 9-es egységen éppen nyitott valamennyi file-t zárja

DEF FN

A használó által meghatározott függvénynek visszaadja az értékét.

DEF FN név (változó) = kifejezés.

Ezzel az utasítással egy komplex számítást mint függvényt lehet meghatározni. Hosszú, a program során többször előforduló képlet használata esetén ezzel a kulcsszóval értékes programhelyet takaríthatunk meg. A függvény neve az FN betűkkel kezdődik, ezt bármilyen, betűvel kezdődő alfanumerikus név követheti. Először a DEF utasítással meghatározzuk a függvényt, ezt követi a függvény neve. A név után zárójelben egy helyettesítő numerikus változónév álljon (ez esetben X), ez után jön az egyenlőségjel, majd a meghatározandó képlet. X helyére bármilyen számot be lehet írni, a következő példa 20-as sorában használt formátum szerint:

Példa:
10 DEF FNA(X) = 12*(34.75–X/.3) + X
20 PRINT FNA(7)
A DEF utasításban megadott képletben az X helyére mindig a 7-es számot teszi a számítógép. A példában a válasz 144.

DELETE BASIC program megadott sorait törli

DELETE [első sor][– utolsó sor]

Ez a parancs csak direkt módban működik.
Példák:

DELETE 75	Törli a 75-ös sort
DELETE 10–50	Törli a 10-estől az 50-es sorig a programot, beleértve a két szélső sort is
DELETE –50	A program elejétől töröl az 50-es sorig (az 50-es sort is beleértve)
DELETE 75–	A 75-ös sortól töröl a program végéig (a 75-öst is)

DIM Megadja egy tömb elemeinek számát

DIM változó(index) [, változó(index)]...

Ha egy tömb 11 elemnél többől áll, a programnak előbb végre kell hajtania a DIM utasítást, hogy a tömb dimenzióit megállapítsa. A DIM utasítás után következik a tömb neve, amely bármilyen legális változónév lehet. Ezt követi zárójelben az egyes dimenziókban található elemek száma (vagy numerikus változója). Az egynél több dimenziójú tömböt mátrixnak nevezzük. Bármennyi dimenziót használhatunk, de ne feledjük, hogy az így létrejött változók listája helyet foglal el a tárban, s az egyhamar elfogyhat. A következőképpen lehet kiszámítani a tömb által elfoglalt tárhelyet:

5 byte a tömb neve;
2 byte minden egyes dimenzió;
2 byte/elem az egész változóknak;
5 byte/elem a normál numerikus változóknak;
3 byte/elem a füzérváltozóknak;
1 byte minden egyes füzérelem minden karaktere számára.

Az egész tömbök a lebegőpontos tömbök helyének ⅖-ét foglalják el (pl. a DIM A%(100) 209 byte-ot, míg a DIM A(100) 512-t).
Egyetlen DIM utasításban egynél több tömböt is lehet dimenzionálni, ha a tömbváltozók nevét vesszőkkel választjuk el egymástól. Ha a program egy tömbnél a DIM utasítást többször végzi el, megjelenik a RE'DIMed ARRAY ERROR hibaüzenet. A programozói gyakorlatban hasznos, ha a DIM utasításokat a program elejére tesszük.

Példa:
10 DIM A$(40),B7(15),CC%(4,4,4)
41 elem, 16 elem, 16 elem

DIRECTORY

Kiírja a képernyőre a lemez tartalomjegyzékének tartalmát.

DIRECTORY [Dmeghajtó száma][<ON,> Ueszközszám][, joker]

C128-as üzemmódban az F3 funkcióbillentyű kiírja a lemez tartalomjegyzékének 8-as eszközszám és 0-s meghajtószám esetén. Ha szüneteltetni akarja a kiírást, használja a CONTROL S vagy a NO SCROLL billentyűket; a szünet után bármelyik billentyűvel újra meg lehet indítani a kiírást. A DIRECTORY paranccsal papírmásolatot nem lehet készíteni, ehhez előbb be kell tölteni a tartalomjegyzéket (LOAD "$",8), tönkretéve ezáltal az éppen a tárban levő programot. Az eszközszám alapértéke 8, a meghajtóé 0.

Példák:

DIRECTORY	A 8-as egységen levő lemez valamennyi file-ját kilistázza.
DIRECTORY D1, U9, "munka"	Kilistázza a „munka" nevű file-t (a 9-es egység 1-es meghajtóján).
DIRECTORY "AB*"	A 8-as egység valamennyi meghajtóján található összes, AB-vel kezdődő file-t kilistázza (pl. ABC, ABODY stb.), mivel a csillag jokert jelent.
DIRECTORY D0, "file ?.BAK"	A ? is joker értékű, bármely azonos helyzetű karakter helyett állhat. Pl. 1.BAK, 2.BAK, 3.BAK stb. mind beleillik a füzérbe.
DIRECTORY D1, U9 (A$)	A 9-es eszközszámon, az 1-es meghajtóban kilistázza az A$ változóban tárolt file-nevet. Ne feledje, hogy amennyiben változót használ filenévnek, azt zárójelbe kell tenni.

Megjegyzés: Ha a 0-s meghajtóban és a 8 egységben levő tartalomjegyzéket ki akarja íratni, kövesse a következő példát:

```
LOAD"S0",8
OPEN4,4,:CMD4:LIST
PRINT#4:CLOSE4
```

DLOAD

BASIC programot tölt be lemezről.

DLOAD "file-név" [,Dmeghajtó száma][,Ueszközszám]

Ez az utasítás BASIC programot tölt be lemezről a mindenkori tárba. (Szalagról való töltéshez használja a LOAD parancsot.) A programot egy max. 16 karakterből álló file-névnek kell jelölnie. A DLOAD a 8-as eszközszámot és a 0-s meghajtószámot tekinti alaphelyzetnek.

Példák:

DLOAD "BUDAPEST"	Megkeresi a lemezen a „BUDAPEST" nevű programot és betölti.
DLOAD (A$)	Az A$ változóban tárolt nevű programot tölti be lemezről. Ha az A$ üres, megjelenik a hibaüzenet. Ne feledje, ha a file-név változó, zárójelbe kell tenni.

BASIC programon belül a DLOAD paranccsal lehet megkeresni egy másik programot a lemezen. Ezt programláncolásnak nevezzük.

DO/LOOP/ WHILE/ UNTIL/EXIT

Meghatározza és vezérli a programciklust.

D0 [UNTIL feltétel/WHILE feltétel] utasítások [EXIT] LOOP[UNTIL feltétel/WHILE feltétel]

Ez a ciklusszerkezet a D0 és a LOOP közötti utasításokat hajtja végre. Ha a D0 és a LOOP utasítást nem módosítja sem az UNTIL, sem a WHILE, akkor a köztük levő utasításokat a számítógép végtelen sokszor fogja végrehajtani. Ha a D0 ciklusba egy EXIT utasítást ágyazunk be, a program átmegy a LOOP utasítást követő első utasításhoz. A D0 ciklusokba beágyazott ciklusokat is lehet írni, a FOR...NEXT szerkezet szabályai szerint. Ha az UNTIL paraméter meg van adva, a program addig hajtja végre a ciklust, amíg (UNTIL) a feltétel igaz nem lesz. A WHILE paraméter alapvetően a UNTIL ellentéte, a programciklus azalatt ismétlődik, mialatt (WHILE) a feltétel igaz. Ha a feltétel már nem igaz, a program a LOOP utáni utasításnál folytatódik. Egy példa a feltételre (logikai paraméter): A = 1 vagy G > 65.

Példák:

```
10 X=25
20 D0 UNTIL X=0
30 X=X–1
40 PRINT "X=";X
50 LOOP
60 PRINT "CIKLUS VEGE"
```

Ez a példa az x = x–1 és a PRINT x = ";x amíg x = 0 utasításokat hajtja végre. Ha y = 0, a program a PRINT "CIKLUS VEGE" utasítással folytatódik, mivel ez áll közvetlenül a LOOP után.

```
10 D0 WHILE A$
= " " :GETKEY A$:LOOP
20 PRINT "A";A$;
"BILLENTYUT NYOMTAD
MEG"
```

A$ mindaddig 0 marad, amíg egyetlen billentyűt sem nyomunk meg. Mihelyt valamelyik billentyűt lenyomjuk, a vezérlés a LOOP utáni PRINT utasításhoz megy. A példa szerint a GETKEY A$ mindaddig működik, amíg A$ karakter. Ez a ciklus állandóan ellenőrzi, hogy megnyomtuk-e valamelyik billentyűt.

```
10 DOPEN#8,
"JATEKOK"
20 D0
30 GET#8, A$
40 PRINT A$;
50 LOOP UNTIL ST
60 DCLOSE#8
```

A program megnyitja a JATEKOK nevű file-t, és addig olvassa be az adatokat, amíg az ST rendszerváltozó nem jelzi, hogy minden adat bevitele megtörtént.

DOPEN

Lemezfile-t nyit meg írásra és/vagy olvasásra.

DOPEN# logikai file-szám, "file-név [,<S/P>]"[,Lfelvétel hossza][,Dmeghajtó száma][<On,> Ueszközszám][,w]

ahol:
S = soros (szekvenciális) file-típus;
P = program file-típus;
L = a felvétel hossza (csak relatív file-ban);
W = írásművelet (ha ezt nem adjuk meg, a gép olvasni fog).
Ezzel az utasítással soros, relatív vagy véletlenszerűen elérhető file-t lehet megnyitni írásra vagy olvasásra. A felvétel hossza (L) relatív file-ra vonatkozik, amely 255 hosszú lehet. A W paramétert csak az írás (PRINT#) művelet közben, soros file esetén kell meghatározni. Ha nem határozzuk meg, a lemezmeghajtó olvasásnak fogja föl a műveletet.
A logikai file-szám a jövőbeni lemezműveletekhez szükséges, pl. olvasás (INPUT#) vagy írás (PRINT#). A logikai fileszám 1 és 255 között bármilyen

szám lehet. A 128-nál nagyobb logikai file-számok automatikusan kocsivisszát és soremelést küldenek minden írási paranccsal. A 128-nál kisebb logikai file-számoknál csak kocsivissza van, amelyet a PRINT# parancs végén pontosvesszővel lehet érvényteleníteni. Az eszköz alapszáma 8, a meghajtóé 0.

Példa:

DOPEN#1, "CIMEK", W	Az 1-es számú soros file-t (CÍMEK) megnyitja írásra.
DOPEN#2 "RECEPTEK", D1, U9	A 2-es számú soros file-t nyitja meg olvasásra (9-es eszköz, 1-es meghajtó.

DRAW

Pontokat, vonalakat és idomokat rajzol a képernyő meghatározott helyére.

DRAW [színforrás], X1, Y1 [TO X2, Y2]...

Ez az utasítás egyedi pontokat, vonalakat és idomokat rajzol. A paraméterek értékei a következők:

Színforrás: 0 bittérképes háttér
1 bittérképes előtér
2 többszínű 1
3 többszínű 2

X1, Y1	Kiindulási koordináta (0,0-tól 320,200-ig)
X2, Y2	Célkoordináta (0,0-tól 320,200-ig)

Az X és Y értékek a pontkurzort helyezhetik abszolút 0 koordinátákba (mint pl. 100, 100) vagy a pontkurzor előző helyéhez viszonyított koordinátába (+/− x és +/− y, pl. +20, −10). Lehet az egyik tengely koordinátája relatív, a másiké pedig abszolút. Az x és y koordináták lehetséges kombinációi a következők:

x,y	abszolút x, abszolút y
+/− x,y	relatív x, abszolút y
x, +/− y	abszolút x, relatív y
+/− x, +/− y,	relatív x, relatív y

Továbbiakat a pontkurzorról a LOCATE parancsnál találhat.

Példák:

DRAW 1, 100, 50	Pontot rajzol
DRAW, 10, 10 TO 100, 60	Vonalat rajzol
DRAW, 10, 10 TO 10, 60, TO 100, 60, TO 10, 10	Háromszöget rajzol

El lehet hagyni paramétert, azonban az azt követő vesszőt ki kell tenni. A meg nem adott paraméterek alapértéket vesznek fel.

DSAVE

BASIC programfile-t lemezre ment ki.

DSAVE "file-név" [,Dmeghajtó száma][<ON,> Ueszközszám]

Ezzel a paranccsal BASIC programot lehet lemezen tárolni (menteni). (A szalagon való tároláshoz l. a SAVE parancsot.) Max. 16. karakterből álló file-nevet kell megadni. Az eszközszám alapértéke 8, a meghajtóé 0.

Példák:

DSAVE "KIADASOK"	A "KIADASOK" nevű programot kimenti lemezre.
DSAVE (A$)	Az A$ változóban tárolt nevű programot menti ki.
DSAVE "PROG 3", D1,U9	A "PROG 3" nevű programot menti ki lemezre, a 9-es egységszámon, 1-es meghajtón.

DVERIFY

Ellenőrzi a tárban a programot és összehasonlítja a lemezen levővel.

DVERIFY "file-név" [,Dmeghajtó száma][<ON, > Ueszközszám]

A számítógép erre a parancsra a megadott meghajtón található programot összehasonlítja a tárban levővel. A meghajtó alapértéke 0, az eszközszámé 8.
Megjegyzés: Amennyiben a SAVE parancs után grafikus területet allokálunk vagy újraallokálunk, hiba keletkezik. Technikailag ez azonban helyes, mert a BASIC szöveg elmozdul az eredeti helyéről, ha bittérképes grafikus területet allokálunk. Így a VERIFY parancs, amely byte-ról byte-ra hasonlít össze, nem működik, bár a program érvényes.
Bináris adatok ellenőrzéséhez l. a VERIFY „file-név", 8,1 formátumot a VERIFY parancs leírásánál.

Példák:

DVERIFY "C128"	A 0-s meghajtón, a 8-as egységen ellenőrzi a "C128" nevű programot.
DVERIFY "SZELLEMEK",D0, U9	A 0-s meghajtón, és a 9-es eszközszámon ellenőrzi a "SZLLEMEK" nevű programot.

END

Definiálja a program végrehajtásának végét.

END

Ha a program az END utasítással találkozik, azonnal abbahagyja a futást. Ha az END után még áll utasítás, a CONT paranccsal onnan lehet folytatni.

ENVELOPE

Definiálja egy hangszer burkológörbéjét.

ENVELOPE n, [,atk][,dec][,sus][,rel][,wf][,pw]

ahol:
n – a burkológörbe száma (0–9),
atk – felfutás (0–15),
dec – lecsengés (0–15),
sus – kitartás (0–15),
rel – elengedés (0–15),
wf – hullámforma: 0 = háromszög,
1 = fűrészfog,
2 = négyszög,
3 = zaj,
4 = gyűrűsmoduláció,
pw – impulzusszélesség (0–4095).

A meg nem adott paraméter megtartja előre definiált vagy újradefiniált értékét. Az impulzusszélesség kizárólag a 2-es számú, négyszög hullámformára vonatkozik, és a következő képlettel határozható meg: pwout = pw/40.95. A Commodore 128-as a következő burkológörbéket inicializálta:

Hangszer	*n*	*A*	*D*	*S*	*R*	*wf*	*pw*
Zongora	0,	0,	9,	0,	0,	2,	1536
Harmonika	1,	12,	0,	12,	0,	1	
Sípláda	2,	0,	5,	15,	0,	0	
Dob	3,	0,	4,	5,	0,	3	
Fuvola	4,	9,	9,	4,	0,	0	
Gitár	5,	0,	9,	2,	1,	1	
Csembaló	6,	0,	9,	0,	0,	2,	512
Orgona	7,	0,	9,	9,	0,	2,	2048
Trombita	8,	8,	9,	4,	1,	2,	512
Xilofon	9,	0,	9,	0,	0,	0	

Ha a megadott hangszereket akarja megszólaltatni, elegendő megadnia a burkológörbe számát, el is hagyhatja a többi paramétert, mivel azok megtartják előre meghatározott értékeiket.

FAST A számítógépet 2 MHz-es üzemmódra állítja.

FAST

Ez a parancs 2 MHz-es módba kapcsol, ezáltal kikapcsolja a VIC 40 oszlopos képernyőt. Minden művelet (az INPUT/OUTPUT kivételével) jelentősen meggyorsul. A grafika használható, azonban a SLOW parancs beviteléig nem lesz látható.

FETCH Adatokat vesz ki a kiterjesztett (RAM modul) tárból.

FETCH# byte-ok száma, intsa, expb, expsa

ahol
a byte-ok száma = a kiterjesztett memóriában fellelhető byteszám (1 – 65 536),
intsa = az eredeti RAM kezdőcíme (0 – 65 535)
expb = a 64 K-s kiterjesztett RAM bankszám (0–3),
expsa = a kiterjesztett RAM kezdőcíme (0–65 535)

FILTER A SID chip hangszűrőjének paramétereit határozza meg.

FILTER [freq][,lp][,bp][,hp][,res]

ahol:
freq = a szűrő levágási frekvenciája;
lp = aluláteresztő szűrő: 0 – ki, 1 – be;
bp = sáváteresztő szűrő: 0 – ki, 1 – be;
hp = felüláteresztő szűrő: 0 – ki, 1 – be;
res = rezonancia (0–15).

A meg nem adott paraméterek esetén az addigi értékek nem változnak. Egyszerre egynél több szűrőtípust is alkalmazhatunk. Pl. az alul- és a felüláteresztő szűrő együttes alkalmazása a sávzáró szűrőt eredményezi. Hogy a szűrőnek hallható eredménye legyen, legalább egyet be kell kapcsolni, és legalább egy szólamot át kell engedni a szűrőn.

Példák:

FILTER 1024,0,1,0,2 A levágási frekvenciát 1024-re állítja be, bekapcsolja a sáváteresztő szűrőt, 2. rezonanciaszinten.

FILTER 2000,1,0,1,10 A levágási frekvenciát 2000-re állítja be, bekapcsolja az alul- és felüláteresztő szűrőket (ezáltal sávzáró szűrőt kapunk); a rezonanciaszint 10.

FOR/TO/STEP/NEXT Ismétlődő programciklust definiál.

FOR változó = kezdőérték TO végérték [STEP növekmény]

Ez az utasítás a NEXT utasítással együtt programciklust alkot, amely meghatározott alkalomszor ismétlődik. Ez akkor hasznos, amikor valamit meg kell számolni, vagy valamit valahányszor meg kell ismételni (pl. nyomtatásnál). Ez az utasítás a FOR és a NEXT utasítások között valamennyi parancsot ismételten végrehajtja, a kezdő- és végértékeknek megfelelően (a ciklusváltozó első

és utolsó száma). A FOR...NEXT ciklus folyamán a ciklusváltozó értéke növekszik vagy csökken.
A FOR...NEXT utasítás logikája a következő: Először is a ciklusváltozó felveszi a kezdeti értéket. Amikor a program eléri a NEXT utasítást tartalmazó sort, hozzáadja a STEP növekményt (alapérték = 1) a ciklusváltozóhoz, és ellenőrzi, hogy nem magasabb-e, mint a ciklus végértéke. Ha kisebb, a ciklus megismétlődik a FOR utasítást követő sortól kezdődően. Ha a ciklusváltozó nagyobb, mint a végérték, a ciklus befejeződik, és a program a NEXT utasítást követő sorban folytatódik. Ha a növekmény értéke negatív, az előbbiek ellenkezője igaz. L. még a NEXT utasítást.

Példa:

```
10 FOR L= 1 TO 10
20 PRINT L
30 NEXT L
40 PRINT "KESZEN VAGYOK! L = "L
```

Ez a program 1-től 10-ig kiírja a számokat, s a végén megjelenik a "KESZEN VAGYOK! L=11" üzenet.
A ciklus végértékét követheti a STEP szó és egy másik szám vagy változó is. Ebben az esetben a STEP után álló érték adódik hozzá a ciklusváltozóhoz 1 helyett. Így törtekkel is lehet számolni, vagy visszafelé, ill. 1-től eltérő értékű növekményekkel.
Arra is mód van, hogy az egyik ciklusba egy másikat helyezzünk el. Vigyázni kell a beágyazott ciklusoknál arra, hogy az utoljára megkezdett ciklus érjen véget először.

Példa:

```
10 FOR L= 1 TO 100
20 FOR A=5 TO 11 STEP .5
30 NEXT A
40 NEXT L
```

A 20-as és 30-as sorokban levő FOR...NEXT ciklus a 10-es és 40-es sorban levőbe van beágyazva. A .5 növekmény azt példázza, hogy a lebegőpontos mutatók érvényesek.

GET

A billentyűzetről bevitt adatokat fogadja, egyszerre egy karaktert anélkül, hogy várna a billentyű lenyomására

GET változólista

A GET utasítással adatokat lehet fogadni a billentyűzetről, egyszerre egy karaktert. Amikor a GET bekerül a programba, a beírt karaktert a számítógép tárolja. Ha nem írunk be karaktert, nulla (üres) karakter jön vissza és a program folytatódik, nem vár billentyűre. A RETURN billentyűt nem szükséges megnyomni. A GET szót egy numerikus vagy füzér változónév kövesse.
Ha a számítógép numerikus billentyűt vár és nem azt nyomunk le, a program megáll, és hibaüzenet jelenik meg. A GET utasítás alkalmazható cikluson belül is, üres eredmény ellenőrzésére. Ebben az esetben a GETKEY-t is lehet használni, l. ott. A GET és a GETKEY utasításokat csak programon belül lehet alkalmazni.

Példa:

10 DO:GETA$:LOOP UNTIL A$ = "A"	Ez a sor azt közli a számítógéppel, hogy csak az A billentyű lenyomására folytassa.
20 GET B, C, D	A B, C és D numerikus változókat fogadja el a billentyűzetről, nem vár arra, hogy egy billentyűt lenyomjunk.

GETKEY Fogadja a billentyűzetről bevitt adatokat, egyszerre egy karaktert, és vár egy billentyű lenyomására.

GETKEY változólista

A GETKEY utasítás sokban hasonlít a GET-hez. Ezzel ellentében azonban a GETKEY azt várja, hogy a használó beírjon egy karaktert. Ezt az utasítást csak programon belül lehet használni.

Példa:

10 GETKEY A$

Ez a sor egy tetszőleges billentyű lenyomására vár, utána a program folytatódik.

10 GETKEY A$,B$,C$

Ez a sor a billentyűzetről bevitt három alfanumerikus karakterre vár.

GET# Szalagról, lemezről vagy az RS232-esről bevitt adatokat fogad.

GET# file-szám, változólista

Ez az utasítás egyszerre egy karaktert visz be egy előzőleg megnyitott file-ról. Egyébként a GET-hez hasonlóan működik. Csak programon belül használható.

Példa:

10 GET# 1, A$ — Ez a sor az 1-es számú file-ról az A$ változóban tárolt egyetlen karaktert fogad, feltételezve, hogy az 1-es file előzőleg már meg volt nyitva. L. az OPEN utasítást.

GO64 C64-es üzemmódba vált.

GO64

Ezzel az utasítással lehet C128-asból C64-es módba váltani. A GO64 utasítás után megjelenik az „Are you sure?" (Biztos?) üzenet. Ha az Y billentyűt nyomjuk meg, az éppen betöltött program elvész és a vezérlés C64-es módba vált át. Ha bármelyik másik billentyűt nyomjuk meg, a számítógép C128-as módban marad. Ez az utasítás egyaránt használható direkt módban vagy programon belül. Ha program módban alkalmazzuk, az üzenet nem jelenik meg.

GOSUB A megadott sorszámnál kezdődő alprogramot (szubrutint) hajtja végre.

GOSUB sorszám

Ez az utasítás hasonló a GOTO-hoz, azzal a különbséggel, hogy a Commodore 128-as visszatér a kiindulási helyére, ha a szubrutinnak vége. Ha RETURN utasítást tartalmazó sorhoz ér a program, a vezérlés visszamegy a GOSUB utasítást követő első utasításra.
A GOSUB utasítás célpontját szubrutinnak (alprogram) nevezzük. Egy szubrutin akkor hasznos, ha egy feladatot a programon belül többször is végre kell hajtani. Ahelyett, hogy a program egy részét állandóan megismételnénk, hozzunk létre szubrutint, és a program megfelelő helyén utaljunk rá egy GOSUB utasítással. L. még a RETURN utasítást.

Példa:

20 GOSUB 800
.
.
.
800 PRINT "SZIA":RETURN

Ez a példa a 800-as sorban kezdődő szubrutint hívja elő és hajtja végre. Minden szubrutin végén RETURN utasításnak kell állnia.

GOTO/GO TO A program végrehajtását a megadott sorszámba viszi át.

GOTO sorszám

Miután a programban GOTO utasítást helyeztünk el, a számítógép a GOTO után álló sorszámú utasítást hajtja végre. Ha direkt módban használjuk, a GOTO a megadott sorszámnál kezdődően hajtja végre a programot anélkül, hogy a változókat törölné. Ez gyakorlatilag ugyanaz, mint a RUN parancs, azzal a különbséggel, hogy nem törli a változó értékeit.

Példák:

10 PRINT "COMMODORE"
20 GOTO 10

A 20-as sorban elhelyezett GOTO utasítás eredményeképpen a 10-es sor folyamatosan ismétlődik mindaddig, amíg a RUN/STOP-ot meg nem nyomjuk.

GOTO 100 Futtatja a 100-as sorban kezdődő programot, de nem törli a változók tárhelyeit.

GRAPHIC Grafikus módba vált.

1) GRAPHIC mód [,clear][,s] vagy
2) GRAPHIC CLR

Ezzel az utasítással a Commodore 128-as a hat grafikus mód valamelyikébe kerül:

Mód	*Leírás*
0	40 oszlopos szöveg
1	Szabvány bittérképes grafika
2	Szabvány bittérképes grafika (osztott képernyő)
3	Többszínű bittérképes grafika
4	Többszínű bittérképes grafika (osztott képernyő)
5	80 oszlopos szöveg

A clear paraméter azt határozza meg, hogy a bittérképes képernyőt a program lefutása után le kell-e törölni (1) vagy érintetlenül kell hagyni (0). Az S paraméter az osztott képernyő kezdő sorszámát jelöli 2-es vagy 4-es grafikus módban (többszínű vagy szabvány bittérképes, osztott képernyős mód). Az osztott képernyő alap kezdőszáma 19.
Végrehajtáskor a GRAPHIC 1–4 9 K bittérképes területet foglal le. A BASIC szövegterület kezdete a bittérképes terület fölött helyezkedik el, és minden BASIC program automatikusan újrafoglalódik. Ez a terület akkor is lefoglalt marad, ha a használó visszatér a szöveg (TEXT) módba (GRAPHIC 0). Ha a törlés paramétere 1, a képernyő törlődik. A GRAPHIC CLR parancs visszafoglalja a bittérképes területet, és ismét elérhetővé teszi a BASIC szöveg számára.

Példák:

GRAPHIC 1,1 Szabvány bittérképes módba vált és törli a bittérképet.

GRAPHIC 4, 0, 10 Osztott képernyős, többszínű bittérképes módba vált, nem törli a bittérképet, az osztott képernyőt a 10-es sorban kezdi.

GRAPHIC 0	40 oszlopos szövegmódba vált.	
GRAPHIC 5	80 oszlopos szövegmódba vált.	
GRAPHIC CLR	Törli és visszafoglalja a bittérképes képernyőt.	

HEADER Lemezt formál.

HEADER „lemeznév" [,Iazonosító][,Dmeghajtó száma]
[<ON , > Ueszközszám]

Mielőtt egy új lemezt először használnánk, a HEADER paranccsal meg kell formálni. Ugyanezzel a paranccsal egy már használt lemezt törölni is lehet, hogy azután újra fel lehessen használni.

Ha a HEADER parancsot direkt módban visszük be, megjelenik az ARE YOU SURE? (Biztos?) üzenet. Program módban ez elmarad.
Ez a program a lemezt blokkoknak nevezett részekre osztja. A file-okról tartalomjegyzéket készít. A lemez neve bármilyen, maximum 16 karakterből álló név lehet. Az azonosító szám tetszőleges két, alfanumerikus karakterből állhat. Jó, ha minden lemeznek egyedi azonosító számot ad. Ügyeljünk a HEADER parancs használatára, mert minden addig tárolt adatot töröl.
Az egyszer már megformált lemezt gyorsabban lehet a HEADER paranccsal formálni, ha nem adunk neki új lemezszámot, hanem a régit használjuk. Ez csak akkor sikerül, ha a lemez előzőleg már volt formálva, mivel ilyenkor csak a tartalomjegyzék törlődik, nem pedig új lemezt formál. Az eszköz alapértéke 8.
A meghajtó számát meg kell adni (szóló lemezmeghajtó esetén ez 0).
Mielőtt a számítógép elvégezné a műveletet, a rendszer elővigyázatosságból megkérdi, hogy "ARE YOU SURE?" (Biztos?). Ha végre akarja hajtani a HEADER parancsot, nyomja meg az Y billentyűt, ha nem, bármelyik másikat.
A HEADER parancs olvassa a lemezparancsok hibacsatornáját és ha hibát talál, megjelenik a "?BAD DISK ERROR" (?Rossz lemez, hiba) üzenet.
A HEADER parancs azonos a következő BASIC 2.0 paranccsal:

OPEN 1, 8, 15, "N0:lemeznév, azonosító"

Példák:

HEADER "DANI", I23, D0	A „Dani" nevű lemezt formálja az I23-as azonosítószámon, a 0-s meghajtón, a 8-as eszközszámon.
HEADER "SZAMOK", I45. D1, ON U9	Megformálja a "SZAMOK" nevű lemezt az I45-ös azonosítószámon, 1-es meghajtón, 9-es eszközszámon.
HEADER "C128 PROGRAMOK", D0	Ez gyorsformáló a 0-s meghajtón, 8-as eszközszámon, feltéve, hogy a lemezt már előzőleg megformáltuk, és a régi azonosítószámot használjuk.
HEADER (A$),I(B$),D0, U9	Ez a sor az A$ változónevű lemezt formálja, az azonosító számát a B$ jelzi, a 0-s meghajtón, 9-es eszközön.

HELP Rámutat a hibás sorra.

HELP

Akkor használjuk, ha a program hibaüzenetet küld. Ha 40 oszlopos formátumban írjuk be a HELP-et, a hibás sort a számítógép kilistázza, a hibát pedig inverzben jeleníti meg. 80 oszlopos formátumban a hibás rész aláhúzva jelenik meg.

IF...THEN/ELSE Feltételes állítást értékel és a program részeit az eredménytől függően hajtja végre.

IF kifejezés THEN utasítások [:ELSE záradék]

Az IF...THEN utasítás BASIC állítást értékel és a két lehetséges megoldás közül az eredménytől függően kiválasztja az egyiket. Ha az állítás igaz, a THEN-t követő utasítást (utasításokat) hajtja végre. Ez bármilyen BASIC utasítás lehet. Ha az állítás hamis, a program az IF-et tartalmazó sor utáni első sorban folytatódik, ha nincs megadva ELSE záradék. Az egész IF...THEN utasítás nem lehet 160 karakternél hosszabb. L. még a BEGIN/BEND utasítást.
Amennyiben ELSE záradék is szerepel a programban, ugyanabban a sorban kell állnia, mint az IF...THEN-nek és a THEN-től kettősponttal kell elválasztani. Az ELSE záradékot csak akkor hajtja végre a számítógép, ha az állítás hamis. Az értéklendő állítás lehet változó vagy képlet, amely akkor igaz, ha nem egyenlő 0-val, és akkor hamis, ha egyenlő 0-val. A legtöbb esetben az állítás relatív operátorokat tartalmaz
(=, <, >, < =, > =, < >).
Az IF...THEN utasításnak két formája lehetséges:
IF kifejezés THEN sorszám
vagy
IF kifejezés GOTO sorszám
Ezek a formátumok a program vezérlését – amennyiben az állítás igaz – a megadott sorszámra viszik át. Ellenkező esetben a program az IF utasítást tartalmazó sort követő sorban folytatódik.

Példa:

```
50 IF X>O THEN PRINT "OK":ELSE END
```

Ez a sor az X értékét ellenőrzi. Ha X nagyobb, mint 0, a THEN kulcsszót követő utasítást hajtja végre a gép (PRINT "OK"), az ELSE-ről nem vesz tudomást. Ha X kisebb vagy egyenlő mint 0, az ELSE záradék valósul meg, és a THEN-t követő utasítást nem veszi figyelembe a számítógép.

```
10 IF X=10 THEN 100
20 PRINT "X nem egyenlő
10-zel"
:
:
99 STOP
100 PRINT "X egyenlő
10-zel"
```

Ez a példa az X értékét méri föl. Ha X=10, a vezérlés a 100-as sorba megy és megjelenik az X=10 üzenet. Ha X nem egyenlő 10-zel, a program a 20-as sorban folytatódik, a gép kiírja az "X nem egyenlő 10-zel" üzenetet, és a program leáll.

INPUT A billentyűzetről adatfüzért vagy számot fogad el, és várja, hogy a használó lenyomja a RETURN billentyűt.

INPUT ["prompt füzér";] változólista

Az INPUT utasítás a program futása közben kér adatokat, és azokat változóhoz vagy változókhoz rendeli. A program megáll, kérdőjelet (?) ír ki a képernyőre, és várja, hogy a használó beírja a választ, és megnyomja a RETURN billentyűt. Az INPUT szót prompt füzér és változónév vagy ezek sora követi, egymástól vesszőkkel elválasztva. A prompt füzér idézőjelbe tett üzenete a használó által beírandó választ sugallja. Az üzenetet követő idézőjel után pontosvesszőnek kell állnia.
Ha egynél több változót olvasunk be, vesszőkkel kell őket elválasztani egymástól. A számítógép két kérdőjel kiírásával kéri a megmaradt értékeket. Ha a RETURN billentyűt megnyomjuk anélkül, hogy értéket bevittünk volna, az INPUT

változó megtartja az előzőleg bevitt értéket. Az INPUT utasítást csak programon belül lehet használni.

Példa:

```
10 INPUT "IRJ BE EGY SZAMOT";A
20 INPUT "ES A NEVEDET";A$
30 PRINT A$ "A/Z/ ";A "SZAMOT IRTAD BE"
```

INPUT# File-ról adatokat visz be a számítógép tárába.

INPUT# file-szám, változólista

Ez az utasítás úgy működik, mint az INPUT, csak egy előzőleg már megnyitott file-tól fogad el adatokat, amelyek nem a billentyűzetről érkeznek, hanem általában lemezen vagy szalagon vannak tárolva. Ezt az utasítást csak programon belül lehet használni.

Példa:

```
10 OPEN 2,8,2
20 INPUT# 2, A$,C,D$
```

Ez az utasítás az A$, C és D$ változókban tárolt adatokat olvassa be a 2-es file-számú, 10-es sorban megnyitott lemezről.

KEY Funkcióbillentyű hozzárendeléseket definiál vagy listáz ki.

KEY [billentyű száma, füzér]

A Commodore 128-as számítógépen nyolc funkcióbillentyű van (F1–F8), négy váltás nélkül használható, négy pedig váltással (a SHIFT billentyű használatával). Valahányszor megnyomunk egy funkcióbillentyűt, a számítógép lehetővé teszi egy funkció vagy művelet elvégzését. A billentyűhöz rendelt definíció állhat adatokból, parancsból vagy parancsok sorozatából. A KEY utasítás, ha nem adunk meg mellé paramétereket, kilistázza az összes billentyűhozzárendelést. Ha adatot rendeltünk a funkcióbillentyűhöz, annak megnyomásakor az adat megjelenik a képernyőn. Az összes definíció együttes hossza max. 246 karakter lehet.

Példa:

```
KEY 7, "GRAPHIC0" + CHR$(13) + "LISTA" + CHR$(13)
```

Ez az utasítás azt közli a számítógéppel, hogy direkt módban az F7 billentyű megnyomásakor kapcsolja be a szövegképernyőt (VIC) és listázza ki a programot. A CHR$(13) a RETURN ASCII karaktere és ugyanúgy működik, mintha megnyomtuk volna a RETURN billentyűt. Az ESCape helyett használja a CHR$(27)-et. A CHR$(34)-gyel kétszeres mennyiségű karaktert lehet egy KEY füzérbe beírni. A billentyűket a program során újra lehet definiálni. Pl.:

```
10 KEY 2,PRINT DS$ + CHR$(13)
```

Ez az utasítás arra szólítja fel a számítógépet, hogy ellenőrizze és írja ki a lemezmeghajtó hibacsatorna változóit (PRINT DS$), valahányszor az F2 billentyűt megnyomjuk.

```
10 FOR L= 1 TO 8:KEY I, CHR$(I + 132):NEXT
```

Ez az utasítás úgy definiálja a funkcióbillentyűket, mint a Commodore 64-esen.

Ha a funkcióbillentyűket BASIC alapértékükre vissza akarjuk állítani, nyomjuk meg a RESET billentyűt.

LET Változóhoz értéket rendel (értéket olvas be a változóba).

[LET] változó = kifejezés

A LET szót ritkán használjuk a programozásnál, mivel használata nem feltétlenül szükséges. Amikor egy változót definiálunk vagy értéket adunk neki, a LET-tel van dolgunk. A változó neve, amely a számítás eredménye lesz, az egyenlőségjel bal oldalán áll, a szám, füzér vagy képlet pedig a jobb oldalán. Egy LET utasítással egyszerre csak egy értéket lehet hozzárendelni a változóhoz. Így pl. a LET A = B = 2 helytelen.

Példa:

10 LET A=5	Az A numerikus változóhoz az 5-ös értéket rendeli.
20 B=6	A B numerikus változónak a 6-os értéket adja.
30 C= A*B + 3	A C numerikus változóhoz az 5-ször 6+3 értéket rendeli.
40 D$ ="SZIA"	A D$ füzérváltozóhoz a "SZIA" füzért rendeli.

LIST Az éppen a tárban levő programot kilistázza.

LIST [első sor][– utolsó sor]

A LIST parancs a számítógép tárába beírt vagy betöltött program listáját jeleníti meg, hogy olvasni és szerkeszteni lehessen. Ha a LIST-et önmagában használjuk, a számítógép az egész programlistát kiírja a képernyőre. A kilistázási folyamatot a C= billentyű lenyomásával lehet lelassítani, a CONTROL S vagy a NO SCROLL billentyűvel lehet szüneteltetni (újra megindítani bármelyik billentyűvel lehet) és a RUN/STOP lenyomásával lehet megállítani. Ha a LIST szó után egy sorszám áll, a számítógép csak azt a sort írja ki. Ha a LIST után két, egymástól gondolatjellel elválasztott szám áll, akkor a két szám közötti összes szám megjelenik. Ha a LIST után egy sorszám és csak egy gondolatjel áll, akkor a megadott sortól a program végéig tekinthetjük meg a listát, végül ha a LIST után gondolatjel és egy szám áll, akkor a program elejétől az illető sorszámig íródik ki a lista. Így a program bármely részét meg lehet vizsgálni vagy változtatni. C128-as módban a LIST-et programon belül lehet használni, és a CONT paranccsal lehet folytatni a programot.

Példák:

LIST	A teljes programot megmutatja
LIST 100–	Listáz a 100-as sortól a program végéig
LIST 10	Csak a 10-es sort mutatja
LIST –100	A program elejétől a 100-as sorig listáz
LIST 10–200	A 10-estől a 200-as sorig írja ki a listát beleértve a 10-es és a 200-as sort is

LOAD A perifériákról, pl. lemezegységről vagy Datassette-ről programot tölt be.

LOAD "file-név" [,eszközszám][,kapcsoló]

Ezzel a paranccsal lehet lemezen vagy magnószalagon tárolt programot behívni. A file-név a program neve, idézőjelben, max. 16 karakterből állhat. A név után vesszőt kell tenni (az idézőjel után) és az eszközszámot, amely arra utal, hogy a program lemezen vagy kazettán van-e. Ha a szám nincs megadva, a számítógép 1-nek veszi az eszközszámot (ez a Datassette kazettás egység száma).

A kapcsoló egy szám, értéke 0 vagy 1 aszerint, hogy hova töltődik be a program a tárban. A 0 esetén a program a BASIC munkaterület elejére töltődik be, ha

értéke 1, a program a tárba oda íródik vissza, ahonnan a SAVE utasítással kimentettük. A kapcsoló alapértéke 0. Az 1-es paramétert általában gépi nyelvű programok töltésénél használjuk.

A LOAD parancsot a leggyakrabban a lemezmeghajtó egység használata mellett alkalmazzuk. Az eszközszám 8, bár lemez esetén ajánlatosabb a DLOAD parancs használata.

Ha a LOAD-ot paraméterek nélkül írjuk be, és közvetlenül a RETURN követi, a számítógép azt feltételezi, hogy szalagról töltünk, és megjelenik a "PRESS PLAY ON TAPE" (nyomd meg a PLAY gombot a magnón) üzenet. Ha ezt teszi, a számítógép a szalagon kezdi keresni a programot. Ha megtalálta, megjelenik a képernyőn a FOUND "file-név" üzenet (megtaláltam a file-t), ahol a file-név az első file neve, amelyet a Datassette a szalagon talál. Ha be akarja tölteni a megtalált file-t, nyomja meg a C= billentyűt, vagy a szóköz billentyűt, ha azt akarja, hogy a gép tovább keressen a szalagon. Ha a program betöltődött, lehet futtatni (RUN), kilistázni (LIST), vagy módosítani.

Példák:

LOAD	A szalagon következő programot beolvassa,
LOAD "HELLO"	megkeresi a szalagon a HELLO nevű programot, és ha megtalálta, betölti.
LOAD A$,8	Lemezről betölti az A$ változóban tárolt nevű programot.
LOAD "HELLO",8	A 8-as meghajtón, 0-s egységen keresi a HELLO nevű programot. (Ez azonos a DLOAD "HELLO"-val.)
LOAD "GEPINYELV",8,1	A GEPINYELV nevű programot arra a helyre tölti be, ahonnan kimentette.

A LOAD parancsot a BASIC programon belül arra is lehet használni, hogy szalagon vagy lemezen a következő programot megkeresse és futtassa. Ezt láncolásnak nevezzük.

LOCATE A bittérképes pontkurzort elhelyezi a képernyőn

LOCATE x,y

A LOCATE utasítás a pontkurzort (PC) a képernyő bármely meghatározott pontkoordinátájára helyezi. A pontkurzor az a koordináta a bittérképes képernyőn, ahol a körök, dobozok, vonalak és pontok rajzolása, valamint kifestése kezdődik. Tartománya a (0,0) x és y koordinátáktól (320,200)-ig terjed. A PC, szemben a szövegkurzorral, nem látható, de a BOX, a CIRCLE, a DRAW stb. grafikus utasításokkal ellenőrizhető. A pontkurzor alaphelyzete az egyes grafikus utasításokban megadott (x,y) koordináta. Így a LOCATE parancsot nem kell specifikálni.

Az x és y értékek a pontkurzort helyezhetik abszolút koordinátákba, pl. (100,100), vagy az előző helyéhez viszonyított, ún. relatív koordinátákba pl. (+20, −10). Az egyik tengely koordinátája lehet relatív, a másiké pedig abszolút. A lehetséges kombinációk a következők:

x,y	abszolút x, abszolút y
+/−x, y	relatív x, abszolút y
x, +/−y	abszolút x, relatív y
+/−x, +/−y	relatív x, relatív y

Példa

LOCATE 160, 100	A pontkurzort a bittérképes képernyő közepére helyezi. Míg valamit nem rajzolunk, a képernyőn semmi sem lesz látható.
LOCATE +20, 100	A pontkurzort a legutolsó PC helytől 20 pontnyira jobbra helyezi, és a 100-as y koordinátába.
LOCATE −30, +20	Az előző PC helyzetéhez képest a pontkurzort 30 pontra balra és 20 pontra lefelé helyezi.

A RDOT(0) funkcióval a PC x koordinátáját, a RDOT(1)-gyel pedig az y koordinátát lehet megtalálni, a pont és a PC színforrását pedig a RDOT(2)-vel.

MONITOR

A Commodore 128-as gépi nyelvű monitorára vált.

MONITOR

A részleteket l. a J) függelékben.

MOVSPR

Sprite-okat helyez el, ill. mozgat a képernyőn.

1) MOVSPR szám, x,y	A megadott sprite-ot az x,y koordinátákba helyezi.
2) MOVSPR szám, +/−x, +/−y	A sprite-ot a pontkurzor helyzetéhez viszonyítva elmozdítja.
3) MOVSPR szám,x;y	A sprite-ot a PC helyzetéhez viszonyítva x távolságra y szögben fordítja el.
4) MOVSPR szám, x szög # y sebesség	A sprite-ot eredeti koordinátáihoz viszonyítva x szögben az óramutató járásával megegyező irányban a megadott y sebességgel mozgatja.

ahol:

a *szám*	a sprite száma (1–8):;
<,x1,y1>	a sprite helyének koordinátái;
a *szög*	az óramutató járásával megegyező irányú szögforgás mértéke (0–360), a sprite-ok eredeti helyéhez képest;
a *sebesség*	a sprite mozgási sebessége (0–15).

Ez az utasítás a képernyő meghatározott helyére helyezi a sprite-okat a SPRITE koordinátasík szerint (ez nem a bittérképes sík), vagy egy bizonyos sebességgel elindítja a sprite-ot. A sprite koordináta-rendszer a 6. alfejezetben a MOVSPR alatt található.

Példák:

MOVSPR 1,150,150	Az 1. sprite-ot a képernyő közepe tájára helyezi, az x,y 150,150 koordinátákba.
MOVSPR 1, +20, −30	Az 1. sprite-ot 20 ponttal jobbra és 30-cal feljebb helyezi.
MOVSPR 4, −50, +100	A 4. sprite-ot 50 ponttal balra és 100-zal lejjebb helyezi.
MOVSPR 5,45#15	Az 5. sprite-ot 45 fokos szögben az óramutató járásával megegyező irányban elforgatja eredeti x,y koordinátáihoz képest. A sprite a maximális sebességgel mozog.

Megjegyzés: Ha a MOVSPR utasítás 3) alakja szerint meghatározzuk a szöget és a sebességet, a többi sprite mozgatása előtt a szöget vissza kell állítani 0-ra, mert befolyásolhatja a mozgásukat.

NEW

Törli a programot és a változó tárat.

NEW

Ez a parancs törli a tárból a teljes programot és valamennyi változót. Ha csak szalagra vagy lemezre ki nem mentettük, a program elveszett. Ezért vigyázzon ennek a parancsnak a használatával. BASIC programban utasításként is lehet használni a NEW parancsot, azonban amikor a számítógép elér ehhez a sorhoz, a programot törli és mindent leállít.

ON Feltételesen leágaztat egy meghatározott programsorba a megadott kifejezés eredményétől függően.

ON kifejezés <GOTO/GOSUB>1. sorszám [,2. sorszám,...]

Ezzel az utasítással a GOTO és a GOSUB utasítások úgy működnek, mint a feltételes IF utasítás speciális változatai. Az ON szót logikai vagy matematikai kifejezés követi, majd vagy a GOTO, vagy a GOSUB és az egymástól vesszőkkel elválasztott sorszámok listája. Ha a kifejezés eredménye 1, a listán elsőként szereplő sorszám utasítását hajtja végre a számítógép. Ha az eredmény 2, a másodikat és így tovább. Ha az eredmény 0, vagy nagyobb, mint ahány szám a listán szerepel, a program az ON utasítás utáni sorban folytatódik. Ha a szám negatív, megjelenik az ILLEGAL QUANTITY ERROR (helytelen mennyiség, hiba) üzenet.

Példa:

```
10 INPUT X:IF X<0
   THEN 10
20 ON X GOTO 30, 40,
   50, 60
30 PRINT "X=1"
40 PRINT "X=2"
50 PRINT "X=3"
60 PRINT "X=4"
```

Ha X=1, az ON a vezérlést a lista első számát viselő sorba küldi (30). Ha X=2, a másodikba stb.

OPEN File-okat nyit meg bevitelre vagy kivitelre (input/output)

OPEN logikai file-szám, eszközszám [,másodlagos cím][,"file-név, file-típus, mód"]/[,cmd füzér]

Az OPEN utasítás segítségével éri el a számítógép a file-okat a lemezegységen, a Datassette-en, a nyomtatón vagy a képernyőn. Az OPEN szót logikai file-szám követi, amely számra hivatkozik a továbbiakban az összes többi BASIC be- és kiviteli utasítás, pl. PRINT# (írj), INPUT# (olvass). A szám a 0–255 intervallumba eshet.

A második szám, az eszközszám. A 0 érték a Commodore 128-as billentyűzetét jelenti, az 1 a magnetofont, a 3 a képernyőt, 4–7 a nyomtató(ka)t és a 8–11 a lemezegysége(ke)t. Hasznos lehet, ha ugyanazt a számot adjuk a file-nak, mint az eszköznek, mert így könnyű emlékezni rá, melyik melyik.

Az eszközszám után állhat egy harmadik paraméter, a másodlagos cím. Szalag esetén ez 0 olvasásnál, 1 írásnál, és 2 írásnál END-OF-TAPE (szalag vége) jelzéssel. Lemez esetén a szám a csatornaszámra vonatkozik. A csatornákról és a csatornaszámokról szóló bővebb információért a lemezegység kézikönyvéhez forduljon. A nyomtatónál a másodlagos címekkel lehet meghatározni bizonyos programozói funkciókat.

A másodlagos címet követheti egy lemezre vagy szalagra specifikált file-név vagy füzér, ez lehet a lemez- vagy kazettaegységnek szóló parancs vagy a file neve. Ha ez utóbbi van megadva, a típus és a mód csak lemezfile-okra vonatkozik. A file-típusok a következők: program-, soros, relatív és felhasználói; a módok: írás és olvasás.

Példák:

10 OPEN 3,3	3-as file-számon megnyitja a képernyőt
20 OPEN 1,0	1-es file-számon megnyitja a billentyűzetet
30 OPEN 1,1,0,"PONT"	Megnyitja a kazettát olvasásra, 1-es file-számon, a file-név PONT
OPEN 4,4	4-es file-számon megnyitja a nyomtatót
OPEN 15,8,15	15-ös file-számon a lemezen megnyitja a parancscsatornát, a másodlagos cím 15. A 15-ös

	másodlagos cím a lemezmeghajtó hibacsatorná-ja számára van fenntartva
5 OPEN 8,8,12, "TESZTFILE, SEQ WRITE"	Soros lemezfile-t nyit meg írásra, amelynek neve Tesztfile, file-száma 8, a másodlagos cím 12

L. még a CLOSE, a CMD, a GET#, az INPUT# és a PRINT# utasításokat, valamint az ST, a DS és a DS$ rendszerváltozókat.

PAINT Egy területet színnel kitölt (befest).

PAINT [színforrás] ,x,y[, mód]

ahol:

a színforrás	0 bittérképes előtér 1 bittérképes háttér 2 többszínű 1 3 többszínű 2
x,y	kezdő koordináta méretezve (alaphelyzet a pontkurzornál)
mód	0 a kiválasztott színforrás által meghatározott színre festi a területet 1 bármelyik, a háttérszíntől különböző színre fest

A PAINT utasítással területet lehet befesteni. Addig festi a megadott pont körüli területet, amíg egy ugyanolyan forrásból származó határral nem találkozik. Pl. ha az előtér színforrásából rajzol kört, ott kezdje a kört kifesteni, ahol a koordináta a háttérszínt feltételezi. A számítógép csak ott fog festeni, ahol a PAINT utasításban meghatározott forrásszín különbözik az x és y pontkoordináták forrásaitól. Olyan pontokat, ahol a PAINT utasításban szereplő forrás és a pontkoordináta forrása ugyanaz, nem tud befesteni. Az x és y koordinátáknak a kifestendő terület határain belül kell elhelyezkedniük, és a kezdő pontkoordináta forrása és a megadott forrásszín eltérő kell hogy legyen.
Az x és y értékek a pontkurzort helyezhetik abszolút koordinátákba (100, 100) vagy az előző helyzethez viszonyított, relatív koordinátákba (+/− x és +/− y) pl. (+20, −10). Az egyik tengely koordinátája lehet abszolút, míg a másiké relatív. A lehetséges kombinációk a következők:

x,y	abszolút x, abszolút y
+/−x,y	relatív x, abszolút y
x, +/−y	abszolút x, relatív y
+/−x, +/−y	relatív x, relatív y

A pontkurzorhoz l. még a LOCATE parancsot.

Példák:

10 CIRCLE 1, 160, 100, 65, 50	Körvonalat rajzol. Kifesti a kört az 1. forrásból (VIC előtér), feltételezve, hogy a (160, 100) pont
20 PAINT 1, 160, 100	háttérszínű (0 forrás.)
10 BOX 1, 10, 10, 20, 20	Megrajzolja egy doboz körvonalát.
20 PAINT 1, 15, 15	Az 1. forrásból kifesti a dobozt, feltételezve, hogy a (15, 15) pont a háttér forrásszíne.
30 PAINT 1, +10, +10	Kifesti a képernyőt az előtér színforrásából, a PC előző helyéhez viszonyított koordinátáihoz képest +10 pontra mind függőleges, mind vízszintes irányban.

PLAY Zenei hangokat és elemeket definiál és játszik le.

PLAY "Vn, On,Tn,Un,Xn,elemek"

ahol:
Vn = szólam (n=1–3)
ON = oktáv (n=0–6)
Tn = burkológörbe alapértékek (n=0–9)
0 = zongora
1 = harmonika
2 = sípláda
3 = dob
4 = fuvola
5 = gitár
6 = csembaló
7 = orgona
8 = trombita
9 = xylofon
UN = hangerő (n=0–15)
Xn = szűrő (n=1 be, n=0 ki)
Hangok: A,B*,C,D,E,F,G

Elemek: # – felemelt hang
$ – leszállított hang
W – egészhang
H – félhang
Q – negyedhang
I – nyolcadhang
S – tizenhatod hang
. – pontozott hang
R – szünet
M – vár, amíg az összes hang kitart az ütem végéig.

A PLAY utasítással meg lehet határozni a szólamot, az oktávot és a burkológörbét (beleértve 10 előre definiált hangszer burkológörbéjét), a hangerőt és a lejátszani kívánt hangokat. A paramétereket idézőjelbe kell tenni. Az R és az M kivételével a PLAY füzér többi eleme megelőzi a zenei hangokat.

Példák:

PLAY "V1O4T0U5X0CDEFGAB"	A C,D,E,F,G,A,B(H) hangokat az 1. szólamban, a 4. oktávon, 0-s burkológörbével (zongorán), 5-ös hangerővel, kikapcsolt szűrővel szólaltatja meg.
PLAY "V3O5T6U7X1# B$AW.CHDQEIF"	Egy gisz, egy asz, egy egész pontozott C, egy fél D, egy negyed E és egy nyolcad F hangot szólaltat meg.

POKE Megváltoztatja egy RAM tárhely tartalmát.

POKE cím, érték

A POKE lehetővé teszi, hogy a Commodore 128-as RAM-jában bármilyen értéket megváltoztassunk, és módosítsunk vele számos C128-as I/O regisztert. A POKE kulcsszó után mindig két paraméter áll. Az első egy hely a Commodore 128-as tárában, értéke 0 és 65 535 között van. A második paraméter értéke a 0 és 255 közti intervallumba esik, ez kerül a tárhelybe, az ott levő érték helyére. A POKE cím a BANK számától függ. L. a kislexikon BANK címszavát.
Példa:
POKE 53280,1 Megváltoztatja a VIC keretszínt.
Megjegyzés: A PEEK, amely a POKE-hoz hasonló funkció, és visszaadja a megadott tárhely tartalmát, a Függvények alatt található.

* Magyarországon H-val jelölik (a fordító).

PRINT Szövegképernyő kivitel (kiírás).

PRINT [kiírandók listája]

A PRINT utasítás a BASIC legfőbb kiviteli utasítása. Ez az első, amelyet a legtöbb használó megtanul, azonban igen sok változata van. A PRINT szó után a következők bármelyike állhat:
- idézőjelbe tett karakterek („szöveg");
- változónevek (A,B,A$,X$);
- függvények (SIN(23), ABS(33));
- írásjelek (; ,).

Az idézőjelbe tett karakterek betűről betűre, pontosan úgy jelennek meg a képernyőn, ahogy beírtuk. A változóknak a tartalma jelenik meg (szám vagy füzér), a függvényeknek szintén a számértéke. Az írásjelek az adatok áttekinthetőségét szolgálják. A vessző 10 üres karakterhelyet jelent, a pontosvessző hármat. Bármelyiket lehet az utasítás utolsó jeleként használni. Ez azt eredményezi, hogy a következő PRINT utasítás olyan lesz, mintha az előző folytatása lenne.

Példák:	Eredmények:
10 PRINT "SZIA"	SZIA
20 A$= "JANCSI":PRINT "SZIA";A$	SZIA JANCSI
30 A = 4:B=2:?A+B	6
40 J=41:PRINT J;PRINT J–1	41 40
50 PRINT A;B;D=A+B:PRINT D;A–B	4 2 6 2

L. még a POS, az SPC, a TAB és a CHAR funkciókat.

PRINT # Adatkivitel file-okba.

PRINT# file-szám, lista

Van néhány különbség a jelen utasítás és a PRINT között. A legfontosabb, hogy a PRINT#-et egy szám követi, amely az előzőleg megnyitott adatfile-ra vonatkozik. A szám után vessző áll, és a file-ra írandók listája. A vessző és a pontosvessző ugyanúgy üres karakterhelyeket biztosít, mint a PRINT utasításnál. Néhány eszköznél előfordul, hogy a TAB-bal és a SPC-szel nem működik.

Példák:

10 OPEN 4,4 20 PRINT#4, "SZIA", A$, B$	A "SZIA"-t, az A$ és a B$ változókat kiviszi a nyomtatóra
10 OPEN 2,8,2 20 PRINT# 2,A,B$,C,D	A 2-es file-számon lemezre viszi az A, a B$, a C és a D adatváltozókat.

Megjegyzés: Ha magában használjuk, a PRINT paranccsal a file bezárása előtt le lehet zárni a nyomtatóhoz menő csatornát az alábbiak szerint:

```
10 OPEN 4,4
30 PRINT#4, "IRJ SZAVAKAT"
40 PRINT#4
50 CLOSE 4
```

PRINT USING Meghatározza a kivitel formátumát.

PRINT [#fileszám] USING "formátumlista"; írandók listája

Ezzel az utasítással a füzér vagy numerikus egységek formátumát lehet szabályozni a szövegképernyőn, nyomtatón vagy más eszközön. A formátumot idézőjelbe kell tenni, ez a formátumlista. Ez után pontosvessző következik, majd azok listája, amiket a megadott formátumban ki akarunk íratni. Ezek lehetnek változók vagy tényleges értékek.

Példa:

```
5 X=32: Y=100.23: A$= "MACSKA"
10 PRINT USING "$##.##";13.25,X,Y
20 PRINT USING "###>#";"CBM",A$
```

Ha ezt futtatjuk, a 10-es sorban ez jelenik meg:

$13.25 $32.00 $*****

Az Y érték helyett azért áll öt csillag, mert az Y öt számjegyből áll és ez a feltétel nem összeegyeztethető a formátumlistával (a magyarázatot l. később).

A 20-as sor ezt írja ki:

CBM MACSKA

A "CBM" előtt három üres karakterhely marad, mint ahogyan a formátumlistában meghatároztuk.

Karakter	*Numerikus*	*Füzér*
# jel	x	x
pluszjel (+)	x	
mínuszjel (−)	x	
tizedespont (.)	x	
vessző (,)	x	
dollárjel ($)	x	
négy beszúrási jel (∧∧∧∧)	x	
egyenlőségjel (=)		x
nagyobb mint jel (>)		x

A # jel egy karakternyi helyet foglal le a kiviteli mezőben. Amennyiben az adatok több karaktert tartalmaznak, mint a formátumban megadott # jelek száma, az egész mezőt csillag tölti ki, egyetlen karakter sem jelenik meg.

Példa:

```
10 PRINT USING "####";X
```

X ezen értékeire a formátum a következőképpen jelenik meg:

A = 12.34	12
A = 567.89	568
A = 123456	****

Füzér esetében a program a füzéradatokat a mező határain levágja. Csak annyi karakter jelenik meg, ahány jel van a formátumban. A levágás a jobb oldalon történik.

A + és − jelek a formátummezőben vagy legelöl vagy leghátul állhatnak, mindkét helyen azonban nem. Ha a szám pozitív, megjelenik a + jel, ha negatív, a − jel. Ha − jelet használunk, a szám pedig pozitív, akkor üres karakterhely jelenik meg a − jel által jelölt helyen.

Ha egy numerikus adatnál a formátummezőben sem plusz, sem mínusz jelet nem használunk, a dollárjel első száma előtt − jel jelenik meg, ha a szám negatív. Ha a szám pozitív, semmilyen jel sem szerepel. Ez azt jelenti, hogy negatív szám esetén egy további karakter, a − jel jelenik meg. Ha túl sok a karakter a # jelekkel és a + jelekkel megadott mezőkhöz képest, a mezőt *-ok töltik meg.

Egy formátummezőben csak egy tizedespont állhat. Ha ezt nem adtuk meg, az érték a legközelebbi egész szám lesz, és tizedeshelyek nélkül jelenik meg. Ha megadjuk a tizedespontot, az azt megelőző számjegyek (beleértve a mínusz jelet is, ha az érték negatív), nem lehetnek többen, mint a tizedespont előtt álló # jelek. Ha túl sok számjegy van, csillagok jelennek meg.

A vessző numerikus mezőkben vesszők elhelyezését teszi lehetővé. A formátumlistában megadott vessző helye jelzi, hol fognak a vesszők a kiírt számban

elhelyezkedni. Csak a számon belüli vesszők jelennek meg. A mezőben az első vessző előtt legalább egy jelnek kell állnia.
Ha egy számban vesszőket adunk meg és a szám negatív, akkor első karakterként mínusz jel szerepel, még akkor is, ha a karakterhely vesszőt jelöl.

Mező	*Kifejezés*	*Eredmény*	*Megjegyzés*
# #. #	−.1	0.1	Az elején hozzáadódik egy 0 (vezető nulla).
# # . #	1	1.0	A végén hozzáadódik egy 0.
# # # #	−100.5	−101	Egészekre kerekítve.
# # # #	−1000	* * * *	Hiba, mert négy számjegy és a mínusz jel nem fér el a mezőben.
# # # .	10	10.	Hozzáadódik a tizedespont.
# $ # #	1	$1	Az elejére hozzáíródik a $ jel.

A dollárjel azt mutatja, hogy a számban $ jel fog megjelenni. Ha a dollárjel lebegő (azaz mindig a szám előtt kell állnia), legalább egy # jelet ki kell tenni a dollárjel elé. Ha a dollárjel elé nem írunk # jelet, akkor oda íródik ki, ahol a formátummezőben megadtuk. Ha vesszők és/vagy mínusz jel áll a formátummezőben a dollárjellel együtt, a program a vesszőt vagy a +/− jelet a dollárjel elé írja ki. A beszúrási jelek (∧ ∧) azt jelölik, hogy a számot E+ formátumban kell kiírni. A mező szélességének jelölésére # jelet kell használni a négy ∧ ∧ ∧ ∧ jel mellett. A beszúrási jelek a # jel előtt és után is állhatnak. Ha a számot E formátumban akarjuk kiírni, négy beszúrási jelet kell megadnunk. Ha egynél többet, de négynél kevesebbet adunk meg, szintaktikai hiba következik be. Ha négynél többet adunk meg, csak az első négy kerül felhasználásra. Az ötödik beszúrási jelet mint nem szöveg szimbólumot értelmezi a számítógép. Az egyenlőségjel a mezőben a füzér középpontját jelöli. A mezőszélességet a karakterek száma adja (# jel és = jel). Ha a füzér kevesebb karaktert tartalmaz, mint a mező szélessége, a füzér középpontja a mezőben helyezkedik el. Ha többet, akkor a jobb szélen álló karakterek levágódnak, és a füzér kitölti az egész mezőt. A nagyobb mint jelet (>) használjuk arra, hogy a füzért jobbra tömörítsük a mezőben.

PUDEF Újradefiniálja a PRINT USING utasítás jeleit.

PUDEF "nnnn"

ahol az "nnnn" tetszőleges karakterkombináció, maximum négy karakterből. A PUDEF-fel az üres karakterhelyeket, vesszőket, tizedespontokat és dollárjeleket lehet újradefiniálni. E négy jelet más karakterrel lehet helyettesíteni, ha a PUDEF füzérbe a megfelelő helyre tesszük őket.
Az 1. pozíció a kitöltő karakter. Az alapérték az üres karakterhely. Az üres karakter helyére bármilyen új karakter beírható.
A 2. pozíció a vessző karakter. Az alapérték a vessző.
A 3. pozíció a tizedespont. Ez az alapérték.
A 4. pozíció a dollárjel. Ez az alapérték.

Példák:
10 PUDEF "*" *-okat ír ki az üres helyekre.
20 PUDEF " <" < jeleket ír ki a vesszők helyére.

READ Adatokat olvas be a DATA utasításokból és beteszi őket a számítógép tárába (a program futása közben).
READ változólista

Ez az utasítás a DATA utasítások információit tárolja változókban, ahol az adatokat a program futtatásával lehet felhasználni. Az utasításban szereplő változólista egyaránt tartalmazhat füzéreket és számokat. Vigyázzon, hogy ne füzéreket olvastasson be ott, ahol a READ utasítás számokat vár és megfordítva. Ekkor ui. megjelenik a TYPE MISMATCH hibaüzenet.

A DATA utasítás adatait a számítógép sorrendben olvassa be. Minden utasítás egy, vagy egynél több adatot képes beolvasni. Az utasítás minden változója adatot igényel. Ha ezt nem adtuk meg, megjelenik az OUT OF DATA ERROR (kifogytam az adatokból, hiba) üzenet.

Egy programon belül be lehet olvasni az adatokat, majd újraolvasni a RESTORE utasítás segítségével. A RESTORE az adatmutatót az elejére helyezi vissza, ahonnan újra be lehet olvasni az adatokat. L. a RESTORE utasítást.

Példák:

10 READ A, B, C 20 DATA 3, 4, 5	Beolvassa a legközelebbi DATA utasításból az első három numerikus változót.
10 READ A$, B$, C$ 20 DATA JANCSI,PALI, MARI	Beolvassa a legközelebbi DATA utasításból az első három füzérváltozót.
10 READ A,B$,C 20 DATA 1200, ABC, 345	Beolvas (és tárol a számítógép tárában) egy numerikus változót, egy füzérváltozót és egy másik numerikus változót.

RECORD Relatív file mutatók elhelyezésére szolgál.

RECORD logikai file-szám, rekordszám [,byte-szám]

Ez az utasítás a relatív file mutatót úgy állítja be, hogy a relatív file-ban bármelyik byte-ot (karaktert) bármelyik rekordról ki lehessen vele választani. A logikai file-szám tartománya 0-tól 255-ig terjed. A rekordszámé 0-tól 65 535-ig. A byte-szám 0 és 254 közötti érték. A relatív file-okról bővebben l. a lemezegység kézikönyvét.

Ha a rekordszám értéke magasabb, mint a file-ban levő legutolsó rekordszám, a következő történik:

Írásműveleteknél (PRINT #) további rekordok keletkeznek, hogy a file a kívánt rekordszámra kibővüljön.

Olvasásnál (INPUT #) zéró rekordot kapunk és megjelenik a "RECORD NOT PRESENT ERROR" (nincs rekord, hiba) üzenet.

Példák:

```
10 OPEN 2,8,2"PROBA,R,W,"
20 RECORD #2,10,1
30 PRINT # 2,A$
40 COLOSE 2
```

Ez az utasítás egy már meglevő, PROBA nevű relatív file-t nyit meg, 2-es file-számon a 10-es sorban. A 20-as sor a relatív file mutatót a 10-es rekordszám első byte-jára helyezi. A 30-as sor pedig a 2-es file-számon beírja az adatokat. A RECORD parancs változókat is elfogad paraméterként. Gyakran hasznos, ha FOR...NEXT vagy DO ciklusba RECORD parancsot helyezünk el. L. még a DOPEN és az OPEN utasításokat.

REM	Egy programsorról megjegyzéseket tartalmaz.	
	REM megjegyzés	
	A REMark (megjegyzés) utasítás a programlista olvasójának szól, megmagyarázhatja a program egy bizonyos részét vagy a program készítőjéről közölhet tudnivalókat stb. A REM utasítások nem érintik a programot, mindössze meghosszabbítják, ezáltal tárhelyet foglalva el. A REM kulcsszótól jobbra álló szavakból semmit sem értelmez úgy a számítógép, mint végrehajtandó parancsot, ezért utána ugyanabban a sorban nem állhat semmi olyan utasítás, amelyet végre kell hajtani.	
	Példa: 10 NEXT X:REM ez a sor megnöveli az X értékét.	
RENAME	Lemezen megváltoztatja a file nevét.	
	RENAME "régi file-név" TO "új file-név" [,Dmeghajtó száma][,Ueszközszám]	
	Ez a parancsot file-ok átnevezésére használjuk. Nyitott file-t nem nevez át a meghajtó.	
	Példák:	
	RENAME "TESZT" TO"PROBLEMAK" ,D0	A TESZ nevű file nevét PROBLEMAK-ra változtatja.
	RENAME (A$) TO (B$) ,D0,U9	Az A$-ban megadott nevet B$-ra változtatja A 0-s meghajtón, 9-es eszközszámon. Ne feledjük, hogy a változó nevét zárójelbe kell tenni.
RENUMBER	Újraszámozza a BASIC program sorait.	
	RENUMBER [új kezdő sorszám][, növekmény][, régi kezdő sorszám]	
	Az új kezdősor a program átszámozása utáni első sor száma. Az alapérték 10. A növekmény alapértéke szintén 10. A régi kezdőszám az átszámozás előtti első sorszám. Arra is lehetőség van, hogy a programnak csak egy részét számozzuk át. Ilyenkor az alapérték a program első sora. A parancs csak direkt módban használható. Ha nemlétező sorszámra történik hivatkozás, megjelenik a LINE NUMBER NOT FOUND ERROR hibaüzenet, amikor pedig a sorszámozás a program keretein túlmegy, az OUT OF MEMORY hibaüzenet. A programnak egyik esetben sem történik baja.	
	Példák:	
	RENUMBER	A 10-estől kezdődően átszámozza a programot, minden sornak 10-zel nagyobb sorszámot adva.
	RENUMBER 20,20,1	Az 1-es sorral kezdődően átszámozza a program sorait. A régi 1-es sor lesz a 20-as, a növekmény értéke szintén 20.
	RENUMBER, , 65	A 65-ös sornál kezdve 10-es növekménnyel újraszámozza a program sorait. A 65-ös lesz a 10-es sor. Ha elhagyunk egy paramétert, a vesszőt azért ki kell tenni a helyére.
RESTORE	Visszaállítja a READ mutatót, hogy az adatokat ismét be lehessen olvasni.	
	RESTORE [sorszám]	

Programban a DATA utasítás következő elemének mutatója az első elemre áll vissza, így az adatokat újra be lehet olvasni. Ha a RESTORE utasítást sorszám követi, a READ mutató a megadott programsor első adatjára kerül. Egyébként a mutató a BASIC program elejére kerül.

Példák:

```
10 FOR I = 1 TO 3
20 READ X
30 TOTAL = X + TOTAL
40 NEXT
50 RESTORE
60 GOTO 10
70 DATA 10,20,30
```

Ez a példa leolvassa a 70-es sor adatait és az X numerikus változóban tárolja. Összeadja az összes numerikus adatot. Ha minden adatot beolvasott a ciklus során, a mutató a program elejére kerül vissza, a 10-es sorba és ismételten végrehajtódik.

```
10 READ A,B,C
20 DATA 100,500,750
30 READ X,Y,Z
40 DATA 36,24,38
50 RESTORE 40
60 READ S,P,Q
```

Ez a példa a DATA mutatót a 40-es sor kezdő adatelemére helyezi vissza. A 60-as sor végrehajtásakor a 40-es sor DATA 36,24,38 adatait olvassa be, mert a 20-as sor adatait már nem kell még egyszer beolvasni.

RESUME Meghatározza, hol folytatódik a program, miután megtalálta a hibát.

RESUME [sorszám/NEXT]

Ezt az utasítást arra használjuk, hogy újraindítsunk vele programot, miután a hibát megtaláltuk (TRAP). Ha nem adtunk meg paramétereket, a RESUME parancs újra végre akarja hajtani azt a sort, amelyikben a hiba előfordult. A RESUME NEXT a hibás sort tartalmazó első utasításnál kezdi meg a program végrehajtását; a RESUME és egy sorszám esetén a megadott sortól folytatódik a végrehajtás. A RESUME utasítást csak program módban lehet használni.

Példa:

```
10 INPUT "◘írj be egy számot";A
15 TRAP 100:B=100/A
40 PRINT "AZ EREDMENY= ";B: PRINT "VEGE"
50 INPUT "AKAROD-E UJRA FUTTATNI (I/N)";Z$:IF Z$="I" THEN 10
60 STOP
100 INPUT "IRJ BE MEG EGY SZAMOT (NEM NULLAT)";A
110 RESUME 15
```

Ebben a példában kiszűrjük a nullával való osztás hibáját a 15-ös sorban, amennyiben a 10-es sorban 0-t írtunk volna be. Ha ez történt, a program elmegy a 100-as sorba, és megkér, hogy írjunk be egy nullától különböző másik számot. A 110-es sor visszatér a 15-öshöz, hogy a számítást befejezze. Az 50-es sor azt kérdezi meg, hogy kívánjuk-e újra futtatni a programot. Ha igen, nyomjuk meg az I billentyűt.

RETURN Visszatérés szubrutinból.

RETURN

Ez az utasítás mindig a GOSUB-bal együtt áll. Ha a program a RETURN utasítással találkozik, az utolsó végrehajtott GOSUB utasítást követő utasításhoz megy. Ha előtte nem volt GOSUB utasítás, megjelenik a RETURN WITHOUT GOSUB ERROR hibaüzenet, és a program megáll. Minden szubrutin végén RETURN utasítás áll.

Példa:

```
10 PRINT "SZUBRUTIN KEZD"
20 GOSUB 100
30 PRINT "SZUBRUTIN VEGE"
:
:
90 STOP
100 PRINT "1. SZUBRUTIN"
110 RETURN
```

A példában a 100-as sor meghívja a szubrutint, amely kiíratja az "1. SZUBRUTIN" üzenetet, majd a 30-as sorban visszatér (RETURNs) a főprogramba.

RUN BASIC program végrehajtása (futtatása).

1) RUN sorszám
2) RUN "file-név" [,Dmeghajtó száma][,Ueszközszám]

Ha egy programot beírtunk a tárba vagy betöltöttünk, a RUN paranccsal lehet futtatni. A RUN törli a program összes változóját mielőtt végrehajtaná a programot. Ha a RUN parancs után egy szám áll, a futtatás attól a sorszámtól kezdődik. Ha a RUN parancs után file-név áll, a megnevezett file betöltődik a lemezegységről és fut. A RUN-t lehet programon belül használni. A meghajtó alapértéke 0, az eszközszámé 8.

Példák:

RUN	A program elejétől kezdi a futtatást.
RUN 100	A 100-as sorban kezdi el a futtatást.
RUN"PRG1"	Lemezről betölti a PRG1 nevű programot és a kezdő sorszámtól futtatja.
RUN(A$)	Az A$ változóban megnevezett programot betölti a lemezről.

SAVE A tárban levő programot kimenti lemezre vagy szalagra.

SAVE ["file-név"][,eszközszám][, lemezvége jelzés]

Ezzel a paranccsal az éppen a tárban levő programot ki lehet menteni szalagra vagy lemezre. Ha a SAVE szót a RETURN követi csak, a számítógép azt feltételezi, hogy a programot magnószalagon kell tárolni. Arra nincsen mód, hogy ellenőrizze, van-e már azon a helyen program a szalagon, így vigyázzunk, nehogy értékes információkra újra felvegyünk valamit. Ha a SAVE-et idézőjelbe tett file-név vagy füzérváltozó követi, akkor a számítógép azon a néven tárolja a programot. Ha eszközszámot is megadunk a SAVE utasításban, akkor az idézőjelbe tett név után írjunk vesszőt, majd egy számot vagy numerikus változót. Az 1-es eszközszám a magnó, a 8-as a lemezmeghajtó. Szalag esetén egy vessző és utána vagy 0, vagy 1 állhat. Ha a második szám 1, a Commodore 128-as END-OF-TAPE (szalag vége) jelzést tesz a program után (csak szalag esetében). Ha egy program betöltésénél a számítógép megtalálja e jelzések valamelyikét, töltés helyett megjelenik a FILE NOT FOUND ERROR hibaüzenet.

Példák:

SAVE	Szalagra menti a programot, név nélkül.
SAVE "SZIA"	Szalagon tárolja a programot, SZIA néven.
SAVE A$,8	Lemezen tárol, az A$ változóban tárolt néven.
SAVE "SZIA",8	Lemezen tárolja a SZIA nevű programot (ez azonos a DSAVE "SZIA"-val).
SAVE "SZIA",1,1	Szalagon tárolja a SZIA nevű programot, és utána szalag vége jelzést tesz.

SCALE — Megváltoztatja a képek méretét grafikus üzemmódban.

SCALE n[,xmax,ymax]

ahol: n = 1 (be) vagy 0 (ki).
standard bittérképes módban:
320 <= Xmax < 32767
(az alapérték = 1023)
200 <= Ymax < 32767
(az alapérték = 1023)
többszínű üzemmódban:
160 <= X max < 32767
(az alapérték = 511)
160 <= Y max < 32767
(az alapérték = 511)

Ez az utasítás megváltoztatja a bittérképek beosztását (méretét) többszínű és nagyfelbontású módokban.
A SCALE 1 beírásával lehet bekapcsolni a méretváltoztatást. Az X és Y koordinátákat ekkor 0-tól 32 767-ig be lehet állítani. A normál értékek a következők:
többszínű üzemmódban X = 0 – 159, Y = 0 – 199
bittérképes üzemmódban X = 0 – 319, Y = 0 – 199.

Példák:

10 GRAPHIC1,1 20 SCALE 1:CIRCLE 1,180,100,100,100	Standard bittérképes módba kapcsol, beállítja az alapértéket (1023,1023) és kört rajzol.
10 GRAPHIC 1,3 20 SCALE 1,1000,5000 30 CIRCLE 1,180,100,100,100	Többszínű üzemmódba vált, a méretet (1000,5000)-re állítja be és kört rajzol.

SCNCLR — Törli a képernyőt

SCNCLR módszám

A módok a következők:

Módszám	Mód
0	40 oszlopos (VIC) szövegképernyő
1	Bittérkép
2	Osztott képernyős bittérkép
3	Többszínű bittérkép
4	Osztott képernyős többszínű bittérkép
5	80 oszlopos szövegképernyő (8563).

Paraméter nélkül ez az utasítás törli a grafikus képernyőt, ha az van jelen, egyébként az aktuális szövegképernyőt törli.

Példák:

SCNCLR 5	Törli a 80 oszlopos szövegképernyőt.
SCNCLR 1	Törli a (VIC) bittérképes képernyőt.
SCNCLR 4	Törli a (VIC) osztott képernyős többszínű bittérképet.

SCRATCH — File-t töröl a lemez tartalomjegyzékéből.

SCRATCH "file-név" [,Dmeghajtó száma][, Ueszközszám]

Ez a parancs egy file-t töröl a lemez tartalomjegyzékéből. Elővigyázatosságból a rendszer megkérdi: ARE YOU SURE? (Biztos?) – csak direkt módban –, mielőtt a számítógép elvégezné a műveletet. Ha a parancsot végre akarja hajtatni, nyomja le az Y billentyűt, ha nem, akkor akármelyik másikat. Ezt a parancsot akkor használjuk, ha egy felesleges file megszüntetésével több helyhez akarunk jutni. A file-névben lehet joker karakter (?, * stb.). A meghajtószám alapértéke 0, az eszközé 8.

Példa:

SCRATCH "HETFO", DO

Ezzel a paranccsal a HETFO nevű file-t lehet kitörölni a 0-s meghajtóban levő lemezről.

SLEEP

Meghatározott időtartamra késlelteti a programot.

SLEEP N

ahol az N másodpercet jelent és az intervallum 0<N<65 535

SLOW

Visszaállítja a Commodore 128-ast 1 MHz-es működési módba.

SLOW

A számítógép a 8502-es mikroprocesszort 1 vagy 2 MHz sebességgel képes működtetni.

A SLOW parancs 2 MHz-ről 1 MHz-re lassítja le a mikroprocesszort. A FAST paranccsal lehet ismét 2 MHz-re beállítani. 2 MHz-es módban a számítógép lényegesen gyorsabb, mint 1 MHz-en.

SOUND

Hanghatásokat állít elő és zenei hangokat szólaltat meg.

SOUND v,f,d [,dir][,m][,s][,w][,p]

ahol:

v = szólam 1–3
f = frekvenciaérték (0–65 535)
d = időtartam (0–32 767)
dir = a hangköz iránya: 0-fel, 1-le, 2-oszcillál. Az alapérték = 0.
m = minimális frekvenciaérték pásztázáskor (0–65 535). Az alapérték = 0.
s = a pásztázás hangköz értéke (0–32 767). Az alapérték = 0.
w = hullámforma: 0-háromszög, 1-fűrészfog, 2-négyszög, 3-zaj. Az alapérték = 2.
p = impulzusszélesség (0–4095). Az alapérték = 2048.

A SOUND paranccsal könnyen és gyorsan lehet hanghatásokat és zenei hangokat előállítani. A három kötelező paraméter, a v, az f és a d beállítja a szólamot, a frekvenciát és a hangzás időtartamát. Az időtartamot jiffy (pillanat) egységekben mérjük. 60 jiffy (másodperc) egyenlő 1 perccel.

A SOUND parancs végigpásztázhat egy sor frekvenciaértéken, ezáltal egy egész hangsoron is. A DIR paraméter határozza meg a pásztázás irányát. Az M határozza meg a minimális értéket, az S pedig a hangköz értékét. A hullámformát a W adja meg, a P pedig az impulzus értékét, amennyiben a W-nek a 2, azaz négyszög hullámforma értéket adtunk.

Példák:

SOUND 1,40960,60 — 1 szólamban, 40 960-as frekvenciájú hangot szólaltat meg 1 másodperc időtartamig.

SOUND 2,20000,50,0,2000,100	A 2000-es frekvenciától kezdődően 100-as növekménnyel folyamatos, pásztázó hangeffektust játszik, minden hangot 50 másodpercig szólaltatva meg.
SOUND 3,5000,90,2,3000,500,1	Ebben a példában egy sor hang szólal meg, 3000-es minimális frekvenciaértéktől kezdődően 5000-ig, 500-as növekményértékkel. Az irány oszcillál. A kiválasztott hullámforma a fűrészfog, a 3. szólamban.

SPRCOLOR Minden sprite-nak többszínű 1 és/vagy többszínű 2 értéket ad.

SPRCOLOR [smcr–1][,smcr–2]

ahol:
smcr–1 többszínű 1 mód minden sprite számára,
smcr–2 többszínű 2 mód minden sprite számára.

A paraméterek bármelyike az 1–16 intervallumban bármelyik szín lehet.

Példák:

SPRCOLOR 3,7	A többszínű 1 sprite-ot pirosra, a többszínű 2-t kékre állítja be.
SPRCOLOR 1,2	A többszínű 1 sprite-ot feketére, a többszínű 2-t fehérre állítja be.

SPRDEF *SPR*ite *DEF*iniáló módba vált, hogy sprite képeket lehessen szerkeszteni.

SPRDEF

SPRDEF módban megjelenik a sprite munkaterület a képernyőn, amely 24 karakterhely széles és 21 karakterhely magas. A rácsozat minden karakterhelye megfelel egy sprite pontnak (pixel), amely a munkaterülettől jobbra elhelyezkedő sprite-ban található. A következőkben ismertetjük a SPRDEF módban használható műveleteket és az azokat végrehajtó billentyűket:

Bevitel	*Leírás*
1–8	Kiválasztja a sprite számát.
A	Ki- és bekapcsolja az automata kurzormozgást.
CRSR billentyűk	A kurzor mozgatásra szolgálnak.
RETURN billentyűk	A következő sor elejére viszi a kurzort.
RETURN billentyűk	A SPRITE NUMBER? üzenetre elhagyja a sprite üzemmódot.
HOME billentyű	A kurzort a munkaterület bal felső sarkába viszi.
CLR billentyű	Törli az egész rácsozatot.
1–4	Meghatározza a színforrást.
CTRL billentyű, 1–8	Kiválasztja a sprite előtérszínét (1/8).
C= billentyű, 1–8	Kiválasztja a sprite előtérszínét (9/16).
STOP billentyű	Törli a változtatásokat és visszatér a bejelentkező üzenethez.
SHIFT RETURN	Kimenti a sprite-ot és visszatér a SPRITE NUMBER? üzenethez.
X	X (vízszintes) irányban növeli a sprite-ot.
Y	Y (függőleges) irányban növeli a sprite-ot.
M	Többszínű sprite.
C	A sprite-adatokat az egyikről a másikra másolja át.

SPRITE Ki- és bekapcsolja, kifesti, nagyítja és beállítja a sprite képernyőprioritásait.

SPRITE <száma>[,be/ki][, fgnd][, prioritás][, x-exp][, y-exp][, mód]

ahol:

szám	a sprite száma (1–8)
be/ki	be- (1) vagy ki- (0) kapcsolja a sprite-ot
fgnd	a sprite előtérszíne (1–16)
prioritás	0, ha a sprite-ok a képernyőn levő tárgyak előtt jelennek meg, 1, ha mögöttük
x-exp	vízszintes kiterjesztés be (1), ki (0)
y-exp	függőleges kiterjesztés be (1), ki (0)
mód	standard sprite- (0) vagy többszínű sprite- (1) módba vált.

A meg nem adott paraméterek a következő sprite utasításoknál az előző Sprite utasítás értékeit veszik fel. A SPRITE jellemzőit a RSPRITE funkcióval lehet ellenőrizni.

Példák:

SPRITE 1,1,3	Bekapcsolja az 1. számú sprite-ot és vörösre festi
SPRITE 2,1,7,1,1,1	Bekapcsolja a 2. sprite-ot, kékre festi, és a képernyőn megjelenő tárgyak mögötti elhaladásra készteti, valamint vízszintes és függőleges irányban kiterjeszti.
SPRITE 6,1,1,0,0,1,1	Bekapcsolja a 6. sprite-ot és feketére festi. Az első 0 azt közli a számítógéppel, hogy a sprite-okat a képernyőn található tárgyak előtt mozgassa. A második 0 és az azt követő 1 arra utasítja a számítógépet, hogy csak vízszintes irányban terjessze ki a sprite-ot. Az utolsó 1 a többszínű üzemmódot definiálja. Használja a SPRCOLOR parancsot, ha többszínű módban akar sprite-okkal dolgozni.

SPRSAV Szövegfüzér változóból sprite-tárba tárol sprite adatokat, vagy fordítva

SPRSAV <forrás>, <cél>

Ezzel a paranccsal sprite-képet lehet füzérváltozóból sprite-tárhelyre menteni, vagy a sprite tárhely adatait át lehet vele vinni füzérváltozóba. A forrás és a cél is lehet egy sprite száma vagy egy füzérváltozó, de mindkettő nem lehet füzérváltozó. Ha füzért viszünk át sprite-ba, akkor az adatoknak csak az első 63 byte-ját használhatjuk. A többiről a számítógép nem vesz tudomást, mivel egy sprite csak 63 adatbyte tárolására képes.

Példák:

SPRSAV 1,A$	Az 1. számú sprite képét átviszi az A$ nevű füzérváltozóba.
SPRSAV B$,2	A B$ füzérváltozó adatait a 2. számú sprite-ba viszi át.
SPRSAV 2.3	A 2. sprite adatait a 3-ba viszi át.

SSHAPE/GSHAPE Füzérváltozókból ment ki és füzérváltozókba tölt be alakzatokat.
A két parancsot többszínű vagy bittérképes képernyőkről négyszögű alakzatok BASIC füzérváltozókba való kimentésére vagy onnan betöltésre használjuk. Az alakzat füzérváltozóba való mentésének parancsa a következő:

SSHAPE füzérváltozó, X1, Y1, [,X2,Y2]

ahol:	
füzérváltozó	A füzér neve, amelybe adatokat akarunk kimenteni.
X1,Y1	A sarok-koordináták (0,0-tól 319,199-ig).
X2,Y2	Az ellenkező sarok koordinátái (az alaphelyzet a pontkurzor).

Mivel a BASIC a füzérek hosszát 255 karakterben maximálja, a kimenthető terület nagysága is korlátozott. A kívánt füzérméret a következők szerint számítható ki:

L(mcm) = INT((ABS(a1 − a2) + 1)/4 + .99)*(ABS(b1 − b2) + 1) + 4
L(h–r) = INT((ABS(a1 − a2) + 1)/8 + .99)*(ABS(b1 − b2) + 1) + 4

Ha füzérváltozóból akarunk adatokat betölteni és meghatározott képernyőkoordináták mellett megjeleníteni, akkor a következő parancsot használjuk:

GSHAPE füzérváltozó[X,Y][,mód]

ahol:	
füzér	Tartalmazza a lerajzolni kívánt alakzatot.
X,Y	Bal felső koordináta (0,0-tól 319,199-ig), amely megmondja, hogy hová rajzolja az alakzatot. Az alapérték a pontkurzor.
mód	Az elmozdítás módja 0– maradjon az alakzat (alapérték), 1– tükrözze az alakzatot (inverz), 2– OR (vagy), 3– AND (és), 4– XOR.

Az elmozdítás módja lehetővé teszi, hogy a füzérváltozóban megváltoztassuk az adatokat, invertálással, logikai VAGY-gyal, kizárólagos VAGY-gyal (XOR) vagy AND (és) művelettel. Az X és Y értékek a pontkurzort helyezhetik abszolút koordinátákba (100,100) vagy az előző helyzethez viszonyított relatív koordinátákba (+20, −10). Az egyik tengely koordinátája lehet relatív, míg a másiké abszolút. A következőkben közöljük az Y és X koordináták lehetséges kombinációit:

x,y	abszolút x, abszolút y
x/ − x,y	relatív y, abszolút y
x, + / − y	abszolút x, relatív y
+ / − x, + / − y	relatív x, relatív y

L. még a LOCATE parancsot a pontkurzorra vonatkozóan.

Példák:	
SSHAPE A$,10,10	Négyszög alakú terület ment ki a (10,10) koordinátákról a pontkurzor helyéig az A$ füzérváltozóba.
SSHAPE B$,20,30,47,51	Négyszöget ment ki (bal felső koordinátája (20,30), jobb alsó (47,51)) a B$ füzérváltozóba
SSHAPE D$, + 10, + 10	A pontkurzor jelenlegi helyétől 10 pontra jobbra és 10 pontra lejjebb ment ki egy négyszög alakú területet.
GSHAPE A$,120,120	Az A$ füzérváltozóban tárolt alakzatot menti ki és a (120,20) bal felső koordinátán megjeleníti.
GSHAPE B$,30,30,1	A B$ füzérváltozóban tárolt alakzatot menti ki és a (30,30) bal felső koordinátában megjeleníti. Az alakzat elfordul, mivel a mód 1.
GSHAPE C$, + 20, + 30	A C$ füzérváltozóban tárolt alakzatot menti ki és a pontkurzor jelenlegi helyétől 20 pontra jobbra és 30 ponttal lejjebb jelenik meg.

Megjegyzés: Ne használjuk az 1–4 módokat többszínű alakzatok esetében, mert nem várt eredményeket kaphatunk.

STASH Az eredeti tár tartalmát a kiterjesztett RAM-ba viszi.

STASH byte-ok száma, intsa, expb, expsa

Lapozza fel a FETCH parancsot, amely a paraméterek leírását tartalmazza.

STOP Megállítja a program végrehajtását.

STOP

Ez az utasítás megállítja a programot. Program módban BREAK IN LINE XXX üzenet jelenik meg, ahol az XXX a STOP parancsot tartalmazó sor száma. A STOP-ot követő utasításoknál újra lehet folytatni a programot, ha közvetlenül utána a CONT parancsot használjuk, anélkül, hogy a listán bármilyen szerkesztésbeli változtatást hajtottunk volna végre. Hiba nyomonkövetésénél gyakran használjuk a STOP utasítást.

SWAP Felcseréli az eredeti RAM tartalmát a kiterjesztett RAM-éval.

SWAP byte-ok száma, intsa, expb, expsa

Lapozza fel a FETCH parancsot, ott megtalálja a paraméterek jelentését.

SYS Gépi nyelvű szubrutint hív és hajt végre egy meghatározott címnél.

SYS cím [,a][,x][,y][,s]

Ez az utasítás egy megadott címnél szubrutint hív be a tárkonfigurációnak a BANK parancsban megadott paraméterei szerint. Az a,x,y és s paraméterek opcionálisan és a megfelelő sorrendben az akkumulátorba, az x-, az y- és a státuszregiszterbe töltődnek a szubrutin meghívása előtt. A cím a 0 és 65 535 közötti intervallumba esik. A program a gépi nyelvű program végrehajtását az ennek megfelelő tárhelynél kezdi. L. még a BANK parancsot.

Példák:

SYS 40960	A 40 960-as tárhelyről behívja és végrehajtja a programot.
SYS 8192,0	A 8192-es tárhelyről behívja és végrehajtja a gépi nyelvű szubrutint, s az akkumulátorba 0-t tölt.

TEMPO Meghatározza a lejátszani kívánt dallam sebességét.

TEMPO n

ahol n egy viszonylagos időtartam 0 és 255 között.

Az egész hang tényleges időtartamát a következő képlet határozza meg: egész hang hangzásának időtartama = 19,22/n másodperc. Az alapérték 8, az időtartam az n-nel növekszik.

Példák:

TEMPO 16	16-ra állítja be a tempó értékét.
TEMPO 1	A tempó értékét a legalacsonyabbra állítja be.
TEMPO 250	A tempó értékét 250-re állítja be.

TRAP BASIC program futása közben nyomon követi és kijavítja a programozásbeli hibákat.

TRAP [sorszám]

Alkalmazásakor a TRAP parancs érzékeli a legtöbb hibafeltételt (kivéve a DOS hibaüzeneteket, viszont a STOP KEY-t igen), kivéve az UNDEF'D STATEMENT ERROR hibaüzenetet. Hiba esetén a hibakapcsoló működésbe lép és a végrehajtás átmegy a TRAP utasításban megjelölt sorba. A hibás sorszámot az EL rendszerváltozó segítségével lehet megtalálni. A hiba körülményeit az ER rendszerváltozó tartalmazza. Az ERR$ (ER) füzérfüggvény megadja a hibák hibaüzenetét.

A program végrehajtását a RESUME utasítással lehet folytatni. Ha nem adunk meg paramétert, a TRAP kikapcsolja a hiba nyomonkövetését. A TRAP rutinban elkövetett hibát nem lehet kinyomtatni. L. még az ST, a DS és a DS$ rendszerváltozókat.

Példa:

100 TRAP 1000	Ha hiba fordul elő, menj az 1000-es sorba.
1000 ?ERR$ (ER);EL	Írd ki a hibaüzenetet és a hibás sorszámot
1010 RESUME	Folytasd a program végrehajtását.

TROFF Kikapcsolja a hiba nyomonkövetését.

TROFF

Ez az utasítás kikapcsolja a hiba nyomonkövető módot.

TRON Bekapcsolja a hiba nyomonkövető módot.

TRON

A TRON-t a program nyomonkövetésére használjuk. Ezzel az utasítással kezdődik a nyomonkövető mód. Ha a programot futtatjuk (RUN), a program sorszámai zárójelben megjelennek, mielőtt bármilyen művelet elkezdődne abban a bizonyos sorban.

VERIFY Összehasonlítja a tárban levő programot a lemezre vagy szalagra kimentett programmal.

VERIFY "file-név" [,eszközszám][,kapcsoló]

Ezzel a paranccsal utasítani lehet a számítógépet, hogy ellenőrizze és hasonlítsa össze a szalagon vagy lemezen tárolt programot a tárban levővel, hogy az valóban ki van-e mentve. Szalag pozicionálására is jól használható, hogy a C128-as a szalagon az utolsó program után kezdje az írást. A gép informálja a használót, ha a programok nem azonosak. Ekkor a szalagot helyesen állítja be és a következő program tárolásánál nem kell attól félni, hogy a régit kitöröljük.
Ha paramétert nem adunk meg, a VERIFY parancs arra használható, hogy a szalagon soron következő programot összevesse a tárban levővel, függetlenül a nevétől. Ha a VERIFY parancsot programnév követi idézőjelben vagy egy füzérváltozó zárójelben, akkor az illető programot kikeresi a szalagon és összehasonlítja a tárban levővel. Ha a VERIFY-t file-név, vessző és egy szám követi, akkor az illető eszközön ellenőrzi a programot (1 – szalag, 8 – lemez). A kapcsoló ugyanaz, mint a LOAD parancs esetében. Arról a tárhelyről ellenőrzi a programot, ahonnan kimentettük.

Példák:

VERIFY	Ellenőrzi a szalagon levő, következő programot.
VERIFY "HELLO"	Megkeresi a szalagon a HELLO-t és összehasonlítja a tárral.
VERIFY "HELLO",8,1	A lemezen megkeresi a HELLO-t és összehasonlítja a tárral.

Megjegyzés: Ha a SAVE után újra lefoglaljuk a grafikus területet, a VERIFY és a DVERIFY parancsra hibaüzenet jelenik meg. Ez technikailag azonban rendben van, mert ebben az esetben a BASIC szöveg átkerült az eredeti (kimentett) helyéről egy másik címtartományba. Így a VERIFY parancs, amely byte-ról byte-ra hasonlít össze, nem működik, még akkor sem, ha a program érvényes.

VOL Meghatározza a hang kimenet erejét.

VOL hangerő szintje

Ez az utasítás a SOUND és a PLAY utasítások számára állítja be a hangerőt, amely 0-tól 15-ig terjedhet. 0-nál kikapcsol. A VOL minden hangra érvényes.

Példák:

VOL 0	A hangerőt a legalacsonyabb értékre állítja (kikapcsolja).
VOL 15	A SOUND és a PLAY utasítások hangerejét a maximumra állítja.

WAIT Szünetelteti a program végrehajtását, amíg egy bizonyos adatfelvétel nem teljesül.

WAIT <hely>, <mask-1> [,<mask-2>]

A WAIT utasítás mindaddig szünetelteti a program végrehajtását, amíg egy megadott tárcím fel nem ismer egy meghatározott bitmintát vagy értéket. Más szóval, a WAIT utasítással késleltetni lehet a programot, amíg egy külső esemény be nem következik. A WAIT-tel bármilyen adatot lehet használni. A legtöbb programozó általában jobb, ha nem használja ezt az utasítást. Csak bizonyos I/O műveleteknél használatos, szinte soha máskor. A WAIT utasítás a tárhelyen levő értéket veszi fel és elvégez egy logikai ÉS műveletet a mask-1 értékével. Ha a mask-2 is meg van adva, az első művelet eredményét VAGYlagosnak veszi a mask-2-vel. Más szóval, a mask-1 „kiszűri" azokat a biteket, amelyeket nem kell ellenőrizni. Ahol a mask-1-ben a bit 0, az eredmény megfelelő bitje is 0 lesz. A mask-2 minden bitet ellenőriz, tehát mind a be-, mind a kikapcsolt (magas vagy alacsony) állapotot meg lehet vizsgálni. A 0-ra vizsgált biteknek a mask-2 megfelelő helyén 1-nek kell lenniük. Ha a mask-1 és a mask-2 operandusok megfelelő bitjei különböznek, a kizárólagos VAGY művelet 1-es bit eredményt ad, ha azonosak, az eredmény 0. A WAIT utasítással végtelen hosszú szünetet is be lehet iktatni, ebben az esetben az újraindításhoz a RUN/STOP és a RESTORE billentyűket kell használni. Előfordulhat, hogy a WAIT-hez BANK parancsot is kell adni, mert az elérni kívánt tár nincs a kiválasztott BANK-ban.
Az alábbi első példában a program addig vár a WAIT utasításra, amíg a szalagegységen meg nem nyomunk egy gombot, s csak akkor folytatódik. A második példánál addig vár, amíg egy sprite nem ütközik a képernyő hátterével.

Példák:
WAIT 1, 32, 32
WAIT 53273, 6, 6
WAIT 36868, 144, 16
(A 144 és a 16 bináris maskok. A 144 bináris alakban 10010000, a 16 pedig 10000.)

WIDTH A rajzolt vonalak szélességét határozza meg.

WIDTH n

Ezzel a paranccsal a BASIC grafikai parancsokban használt vonalak szélességét lehet beállítani egyes vagy kettes szélességre. Ha n-nek 1 értéket adunk, a vonal szimpla szélességű lesz, ha 2-t, akkor kétszeres.

Példák:

WIDTH 1	A grafikai parancsoknál szimplára állítja be a vonalszélességet.
WIDTH 2	A vonalszélesség kétszeres lesz.

WINDOW

Ablakot határoz meg a képernyőn.

WINDOW bal felső oszl, bal felső sor, jobb alsó oszl, jobb alsó sor [,töröl]

Ezzel a paranccsal logikai ablakot lehet készíteni a 40 vagy 80 oszlopos képernyőn. A koordináták tartománya 0-tól 39-ig vagy 79-ig terjed az oszlopérték esetében, és 0-tól 24-ig a soroknál. Ha megadjuk (1) a törlési paramétert, a képernyő letörlődik, de csak az újonnan definiált ablakon belül.

Példák:

WINDOW 5,5,35,20	Ablakot definiál, amelynek bal felső sarka az (5,5), jobb alsó sarka pedig a (35,20) koordinátákra esik.
WINDOW 10,2,33,24,1	Ablakot definiál a (10,2) koordinátájú bal felső és a (33,24) koordinátájú jobb alsó sarokkal, valamint törli is a képernyőnek az ablak által meghatározott részét.

A függvény leírásának formátuma a következő:
FÜGGVÉNY (argumentum)
ahol az argumentum lehet numerikus érték, változó vagy füzér.
Minden függvény leírását példa követi. A példákban kurzívval szedett sorok a számítógép válaszai.

ABS

Visszaadja az abszolút értéket
ABS(X)
Az abszolút érték függvény az X argumentumnak visszaadja a pozitív értékét.

Példa:
PRINT ABS (7*(− 5))
35

ASC

A karakternek visszaadja a CBM ASCII kódját.
ASC(X$)
Ez a függvény az X$ első karakterének ASCII kódját adja vissza. Zéró füzérhez nem kell már CHR$(0)-t adni. Az ILLEGAL QUANTITY ERROR hibaüzenet nem jelenik meg.

Példa:
X$= "C128":PRINT X$
65

ATN

Megadja az X radián tangensű szöget.
ATN(X)
Ez a függvény olyan szöget ad meg, amelynek tangense X, radiánban mérve.

Példa:
PRINT ATN(3)
1,24904577

BUMP

Sprite ütköztetési információt ad.
BUMP (N)
Ha azt akarja megtudni, hogy az utolsó ellenőrzés óta melyik sprite-ok ütköztek, használja a BUMP függvényt. A BUMP(1) azt jelzi, hogy mely sprite-ok ütköztek egymással, a BUMP(2) pedig, hogy melyek a képernyőn található más tárgyakkal. A BUMP-ot a COLLISION-tól függetlenül is lehet használni. A BUMP érték (0–7) bitpozíciói az 1–8 sprite-oknak felelnek meg. Minden használat után a BUMP(n) nullára áll vissza.

A BUMP által megadott érték kettőnek a bitpozícióra emelt hatványa. Pl. ha a BUMP értéke 16, akkor a 4-es sprite ütközött, mert $2^4 = 16$.

Példák:

PRINT BUMP (1)	Azt jelzi, hogy a 2-es és
12	3-as sprite-ok ütköztek.
PRINT BUMP (2)	Azt jelzi, hogy az 5-ös sprite ütközött a képernyő valamely tárgyával.
32	

CHR$

A megadott CBM ASCII kódnak visszaadja az ASCII karakterét.
CHR$(X)
Ez az ASC függvény fordítottja és azt a füzérkaraktert adja meg, amelynek CBM ASCII kódja X. Az E) függelékben közöljük a CHR$ kódok táblázatát.

Példák:

PRINT CHR$ (65) — Megjelenik az A karakter.

A

PRINT CHR$ (147) — Törli a szövegképernyőt.

COS

Az X radián nagyságú szög koszinuszát adja meg.

COS(X)

Ez a függvény X koszinuszának értékét adja meg, ahol az X egy radiánban mért szög.

Példa:

PRIN T COS (π/3)

.5

DEC

A hexadecimális számfüzér decimális értékét adja meg

DEC (hexadecimális füzér)

Ez a függvény a hexadecimális füzér decimális értékét adja meg.

Példa:

PRINT DEC("D020")

53280

ERR$

Valamely hibát meghatározó füzért ad meg.

ERR$(N)

Ez a függvény hibát leíró füzért ad meg. L. még az EL, valamint az ER rendszerváltozókat, továbbá az A) Függeléket, ott közöljük a BASIC hibaüzeneteket.

Példa:

PRINT ERR$(ER)

ILLEGAL QUANTITY ERROR

EXP

e-nek (2.7182813) az X hatványra emelt közelítő értékét adja meg.

EXP(X)

Ez a függvény e-nek (az Euler-féle számnak) (2.7182813) az X-edik hatványát adja.

Példa:

PRINT EXP(1)

2.7182813

FNxx

A használó által meghatározott függvény értékét adja meg.

FNxx(x)

Ez a függvény a DEF FNxx utasításban alkalmazott, a használó által meghatározott xx függvény értékét adja meg.

Példa:

10 DEF FNAA(X) = (X − 32)*5/9

20 INPUT X

30 PRINT FNAA(X)

RUN

? 40 (a ? az input prompt üzenete)

4.44444445

FRE

A tárban elérhető byte-ok számát adja meg.

FRE(X)

ahol X a bankszám. X = 0 BASIC program tárolás esetén, és X = 1, ha az elérhető BASIC változó tárterületét kívánjuk megtudni.

Példák:

PRINT FRE (0) *48893*	A BASIC programok számára felhasználható szabad byte-ok számát adja meg.
PRINT FRE (1) *64256*	A BASIC változók számára fenntartott szabad byte-ok számát adja meg.

HEX$

A decimális számnak a hexadecimális számfüzérét adja meg.
HEX$(X)
Ez a függvény négykarakteres füzért ad meg, amely az X (0<=X 65535) hexadecimális megjelenítését tartalmazza. E függvény decimális megfelelője a DEC.

Példa:
PRINT HEX$(53280)
D020

INSTR

Megadja az 1. füzér helyzetét a 2. füzéren belül.
INSTR (1.füzér, 2.füzér [, startpozíció])
Az INSTR függvény megkeresi a 2.füzér első előfordulását az 1. füzéren belül, és megadja a pozícióját. A startpozíció opcionális paramétere az 1. füzérben adja meg azt a pozíciót, ahol a keresés elkezdődik. Értéke 1 és 255 között lehet. Ha nem találja vagy a startpozíció nagyobb, mint a füzér hossza, vagy ha az 1. füzér nulla, az INSTR függvény 0 értéket ad. Ha a 2. füzér nulla, az INSTR vagy a startpozíció, vagy az 1 értéket adja meg.

Példa:
PRINT INSTR ("COMMODORE 128","128")
11

INT

Egy lebegőpontos érték egész alakját adja.
INT(X)
Ez a függvény egy kifejezés egész értékét adja meg. Ha a kifejezés pozitív, a törtrész marad le. Ha a kifejezés negatív, bármilyen törtrész a hozzá legközelebb álló alacsonyabb egész számot adja.

Példák:
PRINT INT(3.14)
3
PRINT INT(−3.14)
−4

JOY

A joystick helyzetét, valamint a tűzgomb állását adja meg.
JOY(N)
ahol
N=1: a JOY az 1-es számú joystick helyzetét adja meg,
N=2: a JOY a 2-es számú joystick helyzetét adja meg.
128 vagy annál nagyobb érték azt jelenti, hogy a tűzgomb is meg van nyomva. A JOY értéket úgy kereshetjük meg, ha a joystick irányértékéhez hozzáadjuk a 128-at, benyomott tűzgomb esetén. Az irányok a következők:

```
      1
   8     2
7     0     3
   6     4
      5
```

Példák:
JOY(2) = 135
A 2. számú joystick balra tüzel
IF (JOY(1) AND 128) = 128 THEN PRINT "TÜZ"
Meghatározza, hogy meg van-e nyomva a tűzgomb.

LEFT$ — Egy füzér bal szélén álló karaktereit adja meg.

LEFT$ (füzér, egész szám)

Ez a függvény a megadott egész számnak megfelelő karakterből álló füzért ad meg, az eredeti füzér bal szélétől számítva. Az egész számjeggyel megadott argumentumnak a 0 és 255 közötti intervallumba kell esnie. Ha az egész szám nagyobb, mint ahány karakterből a füzér áll, akkor az egész füzért visszakapjuk. Ha az egész szám helyén 0 áll, zéró füzért kapunk eredményül.

Példa:

PRINT LEFT$ "COMMODORE",5

COMMO

LEN — Egy füzér hosszát adja meg.

LEN (füzér)

Ez a függvény a füzérkifejezésben található karakterek számát adja meg. A szóközöket és a ki nem írt karaktereket is beszámítja.

Példa:

PRINT LEN "COMMODORE 128"

13

LOG — X természetes logaritmusát adja meg.

LOG(X)

Ez a függvény X természetes logaritmusát adja meg. A természetes logaritmus az e alapú logaritmus (l. EXP(X)). Ha 10 alapú logaritmusra akarunk átszámítani, osszuk el az eredményt LOG(10)-zel.

Példa:

PRINT LOG 37/5

2.00148

MID$ — Egy hosszabb füzér valamely részét adja meg.

MID$ (füzér, startpozíció [,hossz])

Ez a függvény a hosszúság által meghatározott „alfüzért", részletet ad vissza, a startpozíció által megadott helytől kezdődően. Ennek első karaktere a startpozíció. Az alfüzér hossza a hosszparamétertől függ. Mindkét numerikus argumentum a 0-tól 255-ig terjedő tartományba eshet. Ha a startpozíció értéke nagyobb, mint a füzér hossza, vagy ha a hossz értéke zéró, akkor a MID$ zérófüzér értéket ad meg. Ha nem adjuk meg a hosszparamétert, a startpozíciótól minden jobbra eső karaktert megkapunk.

Példa:

PRINT MID$("COMMODORE 128",3,5)

MMODO

PEEK — Valamely meghatározott tárhely tartalmát adja meg

PEEK(X)

Ez a függvény az X tárhely tartalmát adja meg, ahol X a 0-tól 65 535-ig terjedő intervallumba eshet, az eredmény pedig 0 és 255 közé. Ez a POKE utasítás ellenpárja. Az adatokat a legutolsó BANK utasítás által kiválasztott bankból kapjuk. L. a BANK utasítást.

Példa:

```
10 BANK 15:VIC = DEC ("DOOO")
20 FOR I = 1 TO 47
30 PRINT PEEK (VIC + I)
40 NEXT
```

Ez a példa a VIC chip regisztereinek tartalmát adja meg.

PEN A fényceruza X és Y koordinátáit adja meg.

PEN(n)

ahol

n = 0, a fényceruza helyének X koordinátáját kapjuk;
n = 1, a fényceruza helyének Y koordinátáját kapjuk;
n = 2, a 80 oszlopos képernyő X koordinátáját kapjuk;
n = 3, a 80 oszlopos képernyő Y koordinátáját kapjuk;
n = 4, visszakapjuk a fényceruza tűzgombjának értékét.

Ne feledje, hogy hasonlóan a sprite-koordinátákhoz, a PEN értéke nem méretarányos, és valódi koordinátákat használ, nem pedig grafikus bittérképes koordinátákat. Az X pozíció páros szám, kb. 60-tól 320-ig, míg az Y 50 és 250 között bármelyik szám lehet. Ezek a képernyőn látható koordináta-tartományok, az összes többi érték nem látható a képernyőn. Ha bármelyik pozíción 0 értéket kapunk, azt jelenti, hogy a fényceruza nincs a képernyőn, és az utolsó olvasás óta nem volt megszakítása. Nem fontos, hogy a COLLISION be legyen kapcsolva, ha a fényceruzát használni akarja. Általában kívánatos a fehér háttér használata. A PEN értékek CRT és CRT között változnak.

A 40 oszlopos képernyőtől eltérően a 80 oszlopos (8563) koordináták sor- és oszloppozíciók, nem pedig pontkoordináták, mint a VIC képernyőn. Mind a 40, mind pedig a 80 oszlopos képernyő-koordináták megközelítő értékek, és a fényceruzák természetétől függően változnak. A leolvasott értékek nem érvényesek mindaddig, míg a PEN 4 igaz nem lesz.

Példák:

10 PRINT PEN(0); PEN(1)	Megadja a fényceruza x és y koordinátáit.
10 DO UNTIL PEN(4):LOOP 20 X= PEN(2) 30 Y= PEN(3) 40 REM:REST OF PROGRAM	Biztosítja, hogy a leolvasott értékek érvényesek legyenek.

π Megadja a π értékét

π

Példa:

PRINT π	Ekkor megkapjuk az eredményt: 3.14159265

POINTER Egy változónév címét adja meg.

POINTER (változónév)

Példa:

A = POINTER(Z)	Ez a példa a Z változó címét adja meg.

POS Megadja a kurzor pillanatnyi oszloppozícióját a jelenlegi képernyőablakon belül.

POS(X)

A POS függvény megmutatja, hol van a kurzor a definiált képernyőablakon belül. Az X csak hamis argumentum, meg kell adni, de az értékét a gép nem veszi figyelembe.

Példa:

PRINT POS(0)
10

Kiírja a jelenlegi kurzorpozíciót a meghatározott szövegablakon belül; ezúttal 10.

POT A game paddle potenciométer értékét adja meg.

POT n

ahol, ha
n=1, az 1. számú paddle pozícióját kapjuk;
n=2, a 2. számú paddle pozícióját kapjuk;
n=3, a 3. számú paddle pozícióját kapjuk;
n=4, a 4. számú paddle pozícióját kapjuk.
A POT értékei a 0 és 255 közötti tartományba eshetnek. Ennél nagyobb érték azt jelenti, hogy a tűzgomb is meg van nyomva.

Példa:
```
10 PRINT POT(1)
20 IF POT(1)>256 THEN PRINT"TÜZ"
```
A példa megjeleníti az 1. game paddle értékét.

RCLR A színforrás színét adja meg.
RCLR(N)
Ez a függvény az N forrásszínhez (0<N<6) rendelt színt (1–16) adja meg. Az N értékei a következők lehetnek:

0 = 40 oszlopos háttér;
1 = bittérképes előtér;
2 = többszínű 1;
3 = többszínű 2;
4 = 40 oszlopos keret;
5 = 40 vagy 80 oszlopos karakterszín;
6 = 80 oszlopos háttérszín.

Az RCLR függvény ellentettje a COLOR parancs.

Példa:
```
10 FOR I = 0 TO 6
20 PRINT "FORRÁS";I;"SZINKODJA";RCLR(I)
30 NEXT
```
Ez a példa mind a hat színforrásnak kiírja a színkódját.

RDOT A pontkurzor pillanatnyi helyét vagy színforrását adja meg.
RDOT(N)
ahol:
N = 0 a pontkurzor x koordinátáját adja meg,
N = 1 a pontkurzor y koordinátáját adja meg,
N = 2 a pontkurzor színforrását adja meg.
Ez a függvény a PC (pontkurzor) pillanatnyi helyét vagy színforrását adja meg.

Példák:

PRINT RDOT(0)	a PC x helyzetét adja,
PRINT RDOT(1)	a PC y helyzetét adja,
PRINT RDOT(2)	a PC színforrását adja.

RGR A pillanatnyi grafikus módot adja meg.
RGR(X)
Ez a függvény a pillanatnyi grafikus módot adja meg. Az X hamis (üres) paraméter, azonban meg kell adni. Az RGR függvény ellenpárja a GRAPHIC parancs.
A kiírt érték a következő módokra vonatkozik:

Érték	*Grafikus mód*
0	40 oszlopos (VIC) szöveg
1	Standard bittérkép
2	Osztott képernyős bittérkép
3	Többszínű bittérkép
4	Osztott képernyős többszínű bittérkép
5	80 oszlopos (8563) szöveg.

Példa:

PRINT RCR(0)	Kiírja a pillanatnyi grafikus módot, ez esetben a
1	standard bittérképes kódot.

RIGHT$

Egy füzér jobb széléről adja meg az alfüzért.

RIGHT$ (<füzér>, <szám>)

Ez a függvény a füzér argumentum jobb szélső karakterétől számítva adja meg a füzér egy részletét. Hossza 0 és 255 között bármilyen egész szám lehet, a hosszúság paraméter határozza meg. Ha a szám nulla, akkor zérófüzért kapunk eredményül. Ha a hossz értéke nagyobb, mint a füzér hossza, akkor az egész füzér megjelenik. L. még a LEFT$ és a MID$ függvényeket.

Példa:

PRINT RIGHT$(" BASEBALL",5)

EBALL

RND

Véletlenszámot generál.

RND(X)

Ez a függvény 0 és 1 között generál véletlenszámot. Játékoknál, a kockadobás szimulálására és egyéb véletlenszerű elemek létrehozására használják. Bizonyos statisztikai eljárásoknál is alkalmazható.

Ha X = 0,	az RND a hardver óráján alapuló véletlenszámot ad.
Ha X > 1,	az RND a magértéken alapuló (l. alább), reprodukálható pszeudo-véletlenszámot ad.
Ha X < 0,	az alapul szolgáló, magnak nevezett véletlenszámot adja.

Ha kockadobást akarunk szimulálni, az INT(RND(1)*6 + 1) képletet kell használnunk. Első lépésként a 0 és 1 közötti véletlenszámot megszorozzuk 6-tal, amely így a 0 és 6 közötti tartományba esik (tulajdonképpen 0-nál nagyobb és 6-nál kisebb lesz). Ekkor hozzáadunk 1-et, így az intervallum 1-nél nagyobb és 7-nél kisebb szélső értékkel bír. Az INT függvény levágja a tizedeseket, s eredményül 1 és 6 közötti egész számot kapunk.

Példák:

PRINT RND(0)	0 és 1 között ír ki véletlenszámot.
.507824123	
PRINT INT(RND(1)*100 + 1)	1 és 100 között ad véletlenszámot.
89	

RSPCOLOR

Sprite többszínű értéket ad.

RSPCOLOR (regiszter)

ha X = 1, akkor a többszínű 1 sprite-ot adja,
X = 2, akkor a többszínű 2-t.

A kapott érték 1 és 16 közötti szám. Az RSPCOLOR ellenpárja a SPRCOLOR utasítás, l. ott is.

Példa:

```
10 SPRITE 1,1,2,0,1,1,1
20 SPRCOLOR 5,7
30 PRINT "SPRITE TOBBSZINU 1 = ";RSPRCOLOR(1)
40 PRINT "SPRITE TOBBSZINU 2 = ";RSPRCOLOR(2)
RUN
SPRITE TOBBSZINU 1 = 5
SPRITE TOBBSZINU 2 = 7
```

Ennél a példánál a 10-es sor bekapcsolja az 1. sprite-ot, fehérre festi, mind x mind pedig y irányban megnagyítja, és többszínű üzemmódban megjeleníti. A 20-as sor kiválasztja a többszínű 1 és 2 sprite-okat. A 30-as és 40-es sor kiírja a többszínű 1 és 2 RSPRCOLOR értékeit.

RSPPOS

A sprite sebesség- és pozícióértékeit adja meg.
RSPPOS (sprite száma, pozíciója/sebessége)

ahol a sprite száma az ellenőrzendő sprite azonosítására szolgál, a pozíció és a sebesség pedig a sprite x és y koordinátáit, ill. sebességét jelenti.
Ha a pozíció 0, a megadott sprite jelenlegi x pozícióját kapjuk, ha 1, akkor az y pozíciót.
Ha a sebesség 2, akkor a megadott sprite sebességét (0–15) kapjuk.

Példa:

```
10 SPRITE 1,1,2
20 MOVSPR 1,45#13
30 PRINT RSPPOS(1,0);RSPPOS(1,1);RSPPOS(1,2)
```

A példa a pillanatnyi x és y sprite-koordinátákat, valamint a sebességet (13) adja meg.

RSPRITE

A sprite jellemzőit adja meg.
RSPRITE (sprite száma, jellemzője)
Ez a függvény a SPRITE utasításban meghatározott sprite jellemzőket adja meg. A szám az ellenőrizni kívánt sprite azonosító száma. A jellemzőket a következő táblázat tartalmazza.

Jellemző	*A kapott értékek az RSPRITE nyomán*	
0	Be 1, ki 0	
1	A sprite színe (1–16)	
2	A sprite-ok a tárgyak előtt (0) vagy mögött (1) jelennek meg	
3	x irányban nagyít, igen = 1, nem = 0	
4	y irányban nagyít, igen = 1,	nem = 0
5	többszínű igen = 1,	nem = 0

Példa:

```
10 FOR I = 0 TO 5
20 PRINT RSPRITE(1,I)
30 NEXT
```

Ez a példa az 1. sprite mind a 6 jellemzőjét kiírja.

RWINDOW

A pillanatnyi ablak méretét adja meg.
RWINDOW(n)
ha n = 0, akkor az ablak sorainak számát kapjuk,
n = 1, akkor az ablak oszlopainak számát kapjuk,
n = 2, akkor vagy 40-et vagy 80-at kapunk, a képernyőkivitel formátumától függően.
Az RWINDOW függvény ellenpárja a WINDOW parancs.

Példa:

```
10 WINDOW 1,1,10,10
20 PRINT RWINDOW(0);RWINDOW(1);RWINDOW(2)
RUN
10 10 40
```

A példában az aktuális ablak sorainak számát (10) és oszlopainak számát (10) kapjuk. A példában 40 oszlopos képernyőformátumot használtunk.

SGN Az X argumentum (független változó) előjelét adja meg.

SGN(X)

Ez a függvény X-nek az előjelét (pozitív, negatív vagy 0) adja meg. Az eredmény +1, ha X>0; 0, ha X=0 és −1, ha X<0.

Példa:

PRINT SGN(4.5);SGN(0);SGN(−2.3)

1 0 −1

SIN Az argumentum szinuszát adja meg.

SIN(X)

Ez a trigonometriai szinuszfüggvény. Az eredmény X-nek a szinusza. X-et radiánban mérjük.

Példa:

PRINT SIN(π/3)

.866025404

SPC Üres karakterhelyeket hagy ki a képernyőn.

SPC(X)

Ezt a függvényt a PRINT vagy a PRINT# parancsoknál használjuk az adatok formátumának meghatározására vagy képernyőkivitelnél, vagy logikai file kivitelnél. Az X által determinált üres karakterhelyek száma adja a képernyőn vagy a file-ban az üresként megjelenő karaktereket. Képernyő- vagy szalagfile-ok esetében az argumentum értéke a 0 és 255 közé esik, lemezfile-oknál a maximum 254. Nyomtató file-oknál automatikus kocsivissza és soremelés történik, ha üres karakterhelyet írunk be egy sor utolsó karakterpozíciójába. A következő sorban nem lesznek üres karakterhelyek.

Példa:

PRINT "COMMODORE";SPC(3);"128"

COMMODORE 128

SQR Az argumentum négyzetgyökét adja

SQR(X)

Ez a függvény X négyzetgyökének értékét adja meg, ahol X pozitív szám vagy 0. A változó értéke nem lehet negatív, ebben az esetben ui. az ?ILLEGAL QUANTITY hibaüzenet jelenik meg.

Példa:

PRINT SQR(25)

5

STR$ Egy szám füzéralakját adja meg.

STR$(X)

Ez a függvény az X független változó numerikus értékének füzérét jeleníti meg. Amikor az STR$ értékét átszámítjuk a változó valamennyi értékére, a kiírt számok előtt és után üres karakterhely jelenik meg, kivéve a negatív számokat, amelyek előtt mínusz jel áll. A STR$ függvény ellenpárja a VAL függvény.

Példák:

PRINT STR$(123.45)

123.45

PRINT STR$(−89.03)

−89.03

PRINT STR$(1E20)

1E+20

TAB

A kurzort a jelenlegi utasítás tabulátorhelyzetére viszi.

TAB(X)

Ez a függvény a kurzort, amennyiben lehetséges, előre viszi az X független változó által meghatározott relatív helyzetbe, a sor bal széléről kezdődően. A változó értéke a 0–255 intervallumba eshet. Ha az aktuális PRINT pozíció már túl van az X helyzeten, a TAB függvény a kurzort a következő sor X helyére viszi. A TAB függvényt csak a PRINT utasítással lehet használni, mert hatástalan marad a PRINT# logikai file-ba utasítással szemben.

Példa:

10 PRINT"COMMODORE"TAB(25)"128"

COMMODORE 128

TAN

A független változó tangensét adja meg.

TAN(X)

Ez a függvény X-nek a tangensét adja, ahol X egy radiánban mért szög.

Példa:

PRINT TAN(.785398163)

1

USR

Lehívja a használó által definiált alprogramot.

USR(X)

Ha ezt a függvényt használjuk, a program egy olyan gépi nyelvű programra ugrik, amelynek kezdőpontja a 4633-as ($1219) és a 4634-es ($121A) tárhelyeken van, C64-es mód esetében pedig ugyanaz a 785($0311) és a 786($0312). Az X paraméter a gépi nyelvű program lebegőpontos akkumulátorába kerül. A változó segítségével lehet az értéket visszavinni a BASIC programba. A programban vissza kell állítani az értéket változóvá, hogy visszakapjuk a lebegőpontos akkumulátorból. Ha ezt a változót nem adjuk meg, megjelenik az ILLEGAL QUANTITY ERROR hibaüzenet. Ekkor a használó kicserélheti a változót a gépi kód és a BASIC között.

Példa:

10 POKE 4633,0
20 POKE 4634,192
30 A = USR(X)
40 PRINT A

A gépi nyelvű rutin kezdőhelyét ($COOO = 49152:$OO = O:$CO = 192) a 4633-as és a 4634-es tárhelyre tegyük. A 30-as sor a lebegőpontos akkumulátorból kapott értéket tárolja.

VAL

Számfüzér numerikus értékét adja meg.

VAL(X$)

Ez a függvény az X$ füzért számmá alakítja át, ez a STR$ ellentétes művelete. A számítógép a füzér bal széléről kiindulva halad jobbra; annyi karaktert vizsgál meg, amennyi a felismerhető numerikus formátumban van. Ha a gép illegális karaktereket talál, akkor csak addig a pontig számítja át a füzért. Ha nincsenek numerikus karakterek, a VAL függvény eredménye 0 lesz.

Példa:

10 A$ = "120"
20 B$ = "365"
30 PRINT VAL A$ + B$
RUN

485

XOR Kizárólagos VAGY-ot ad meg.

XOR(n1,n2)

Ez a függvény az n1 és n2 független változók értékének kizárólagos logikai VAGY eredményét adja.

x = XOR(n1,n2)

ahol n1 és n2 tetszőleges értékek (0–65535)

Példa:

```
PRINT XOR(128,64)
```

192

19. ALFEJEZET

Változók és operátorok

Változók 188

Operátorok 188

VÁLTOZÓK

A Commodore 128-as háromféle változót használ BASIC nyelven. Ezek a következők: normál numerikus, egész numerikus és füzér (alfanumerikus).

A normál NUMERIKUS VÁLTOZÓKNAK (lebegőpontos változóknak) −10-től +10-ig bármilyen kitevő értékük lehet, 9 jegy pontossággal. Ha egy szám nagyobb, mint amit kilenc számjeggyel ki lehet fejezni, mint pl. +10 vagy −10 esetén, a számítógép tudományos (normál) alakban írja ki, egy egész és nyolc tizedes hellyel, utána következik egy E betű és a tíz hatványa. Pl. az 12345678901 szám a számítógépen a következő alakban jelenik meg: 1.23456789E+10.

Az EGÉSZ VÁLTOZÓKAT akkor használhatjuk, ha a szám értéke +32767 és −32768 közé esik, és törtrésze nincs. Egész változó pl. az 5, a 10 vagy a −100. Az egészek kevesebb helyet foglalnak el, mint a lebegőpontos változók, különösen ha tömbökben használjuk őket.

A FÜZÉRVÁLTOZÓKAT karakteradatokhoz használjuk. Tartalmazhatnak számokat, betűket és bármilyen más olyan karaktert, amelyet a Commodore 128-as ki tud írni.

A VÁLTOZÓNEVEK állhatnak egyetlen betűből, egy betűből és egy számból vagy két betűből. Hosszabbak is lehetnek két karakternél, de csak az első kettőnek van jelentősége. Az egészet úgy jelöljük, hogy a szám után százalékjel (%) áll, a füzérváltozók neve után pedig dollárjel $.

Példák:
Numerikus változónevek: A, A5, BZ
Egész változónevek: A%, A5%, BZ%
Füzér változónevek: A$, A5$, BZ$

A TÖMBÖK azonos nevű változók listái, a tömb elemének meghatározására külön számot vagy számokat használunk. A tömböket a DIM utasítás segítségével lehet definiálni; lehetnek lebegőpontos, egész- vagy füzérváltozós tömbök. A tömbváltozó neve után zárójelben következik a változó száma.

Példa:
A(7), BZ%(11), A$(87)

A tömböknek egynél több dimenziójuk is lehet. A kétdimenziós tömbben sorok és oszlopok vannak, az első szám a sor, a második pedig az oszlop azonosítására szolgál. (Hasonlóan a térképeknél megadott koordinátarácshoz.)

Példa:
A(7,2), BZ%(2,3,4), Z$(3,2)

A FOGLALT VÁLTOZÓNEVEKET más célra nem lehet használni, mert le vannak foglalva a Commodore 128-as használatára. Ezek a következők: DS, DS$, ER, ERR$, EL, ST, TI és TI$. Hasonlóképpen nem használhatók a kulcsszavak, pl. a TO vagy az IF, és olyan nevek, amelyek kulcsszavakat tartalmaznak pl. RUN, NEW vagy LOAD.

Az ST a bemenet és kimenet státusváltozója (kivéve a normál képernyő és billentyűzeti műveleteket.) Az ST értéke a legutolsó I/O művelet eredményétől függ. Általában véve, ha ST értéke 0, a művelet sikeres volt. A TI és a TI$ változók a Commodore 128-ba épített valós időmérő órára vonatkoznak. A rendszer órája minden 1/60-ad másodpercben felülíródik. 0-nál indít, amikor a gépet bekapcsoljuk, és csak a TI$ értékének megváltoztatásával lehet újraállítani. A TI változó az óra pillanatnyi értékét adja meg, 1/60-ad másodpercben. A TI$ olyan füzér, amely úgy olvassa az óra értékét, mint egy 24 órás normál óra. A TI$ első két karaktere adja az órát, a harmadik és a negyedik a percet, míg az ötödik és hatodik a másodpercet. Ezt a változót bármilyen értékre be lehet állítani (amennyiben a karakterek számok), és automatikusan felülíródik, mint egy 24 órás óra.

Példa:
TI$ = "101530" Az órát 10 óra 15 perc 30 másodpercre állítja be.

Az óra értéke elvész, ha a gépet kikapcsoljuk. Ha bekapcsoljuk, 0-nál kezd, és ismét 0-ra áll be, ha az érték meghaladja a 235 959-et (23 óra 59 perc 59 másodperc).

A DS változó a lemezmeghajtó parancscsatornáját olvassa és a meghajtó pillanatnyi állapotát adja meg. Ha ezt az információt szavakkal akarjuk megkapni, írjuk be a PRINT DS$ utasítást. Ezeket az állapotváltozókat lemezművelet után használjuk, pl. a DLOAD vagy a DSAVE után, hogy megtudjuk, miért villog a lemezmeghajtó hibajelző fénye.

Az ER, az EL, és az ERR$ változókat hibanyomonkövető rutinoknál használjuk. Általában csak programon belül használatosak. Az ER a program futtatása utáni legutolsó hibát adja meg. Az EL azt a sort, ahol a hiba előfordult. Az ERR$ függvény teszi lehetővé, hogy a program valamelyik BASIC hibaüzenetet kiírhassa. A PRINT ERR$(ER) kiírja a megfelelő hibaüzenetet.

OPERÁTOROK

A BASIC operátorok a következők: aritmetikai, relációs és logikai operátorok. Az aritmetikai operátorok jelei:

\+ összeadás
− kivonás
* szorzás
/ osztás
↑ hatványozás.

Ha egy sorban egynél több operátor fordul elő, akkor a számítógép bizonyos meghatározott sorrendben végzi el a műveleteket. A prioritások a következők: első a hatványozás, ezt követi a szorzás és az osztás, végül az összeadás és a kivonás. Amennyiben két operátornak azonos a prioritása, a gép balról jobbra végzi el a számításokat. Ha a számításoknak más sorrendet kívánunk adni, a C128 BASIC nyelve lehetővé teszi, hogy a prioritást zárójelek segítségével megváltoztassuk. Így a zárójelbe került műveletnek lesz a legmagasabb prioritása. Ügyeljünk arra, hogy az egyenletekben azonos legyen a nyitó és záró zárójelek száma, különben megjelenik a SYNTAX ERROR hibaüzenet.

Egyenlőségek és egyenlőtlenségek kifejezésére is vannak operátorok, ezeket relációs operátoroknak nevezzük. Az aritmetikai operátoroknak mindig prioritásuk van a relációkkal szemben.

= egyenlő
< kisebb
> nagyobb
< = vagy = < kisebb vagy egyenlő
> = vagy = > nagyobb vagy egyenlő
< > vagy > < nem egyenlő

Végül vannak logikai operátorok, amelyeknek mind az aritmetikai, mind a relációs operátorokénál kisebb a prioritásuk:

AND (ÉS)
OR (VAGY)
NOT (NEM)

Ezeket általában az IF...THEN utasításokkal együtt használjuk. Ha aritmetikai operátorokkal együtt alkalmazzuk őket, akkor ezeket veszi figyelembe a legutoljára a gép (tehát a + és a − után). Ha a kifejezésben megjelölt viszony igaz, az eredmény a −1 egész érték lesz. Ha hamis, az eredmény 0.

Példák:

IF A = B AND C= D THEN 100	Igaznak kell lennie mind az A=B, mind a C=D-nek.
IF A=B OR C=D THEN 100	Vagy az A=B-nek vagy a C=D-nek kell igaznak lennie.
A=5:B=4:PRINT A=B	0 érték jelenik meg.
A=5:B=4:PRINT A>3	−1 érték jelenik meg.
PRINT 123 AND 15: PRINT 5 OR 7	11 és 7 jelenik meg.

20. ALFEJEZET

Foglalt szavak és jelek

Foglalt rendszerszavak (kulcsszavak) 192

Foglalt rendszerjelek 192

FOGLALT RENDSZERSZAVAK (KULCSSZAVAK)

A következőkben felsoroljuk a BASIC 7.0 nyelv szavait és jeleit. Ezeket nem szabad programon belül másra használni, csak a BASIC nyelv részei lehetnek. Az egyetlen kivétel, amikor PRINT utasításban, idézőjelen belül használjuk őket.

Jel	*Felhasználás*
* (csillag)	Aritmetikai szorzás.
/ (per)jel	Aritmetikai osztás.
↑ (felfelé nyíl)	Aritmetikai hatványozás
Üres karakterhely	a kulcsszavak és változónevek elválasztására szolgál.
= (egyenlőség)	Érték hozzárendelés, viszony ellenőrzése.

ABS	DIM	HEADER	PRINT USING	SPRITE
AND	DIRECTORY	HELP	PRINT#	SPRSAV
APPEND	DLOAD	HEX$	PRINT# USING	SQR
ASC	DO	IF	PUDEF	ST
ATN	DOPEN	INPUT	RCLR	STASH
AUTO	DRAW	INPUT#	RDOT	STEP
BACKUP	D$	INSTR	READ	STOP
BANK	DSAVE	INT	RECORD	STR$
BEGIN	DS$	JOY	RENAME	SWAP
BEND	DVERIFY	KEY	RENUMBER	SYS
BLOAD	EL	LEFT$	RESTORE	TAB
BOOT	ELSE	LEN	RESUME	TAN
BOX	END	LET	RETURN	TEMPO
BSAVE	ENVELOPE	LIST	RGR	THEN
BUMP	ER	LOAD	RIGHT$	TI
CHAR	ERR$	LOCATE	RND	TI$
CHR$	EXIT	LOG	RSPCOLOR	TO
CIRCLE	EXP	LOOP	RSPPOS	TRAP
CLOSE	FAST	MID$	RSPRITE	TRON
CLR	FETCH	MONITOR	RUN	TROFF
CMD	FILTER	MOVSPR	RWINDOW	UNTIL
COLLECT	FN	NEW	SAVE	USR
COLOR	FOR	NEXT	SCALE	VAL
CONCAT	FRE	NOT	SCNCLR	VERIFY
CONT	GET	ON	SCRATCH	VOL
COPY	GETKEY	OPEN	SGN	WAIT
COS	GET#	OR	SIN	WHILE
DATA	GO64	PAINT	SLEEP	WIDTH
DCLEAR	GOSUB	PEN	SLOW	WINDOW
DCLOSE	GOTO	PLAY	SOUND	XOR
DEC	GO TO	POS	SPC	
DEF FN	GRAPHIC	POT	SPRCOLOR	
DELETE	GSHAPE	PRINT	SPRDEF	

FOGLALT RENDSZERJELEK

A következő karakterek a rendszer foglalt jelei:

Jel	*Felhasználás*
+ (plusz)	Aritmetikai összeadás, füzér kapcsolás, relatív pontkurzor mozgatás, gépi nyelvű monitoron tizedest jelöl.
− (mínusz)	Aritmetikai kivonás, negatív szám, unáris mínusz, relatív pontkurzor/sprite mozgatás.
< (kisebb, mint)	Viszony kifejezése.
> (nagyobb, mint)	Viszony kifejezése.
, (vessző)	A változólistáknál a kivitel formátumához, parancs, ill. utasítás függvény paraméterei.
. (pont)	Lebegőpontos állandóknál tizedespont.
; (pontosvessző)	Változólistáknál a kimenet formálására.
: (kettőspont)	Egyazon programsorban elválasztja egymástól a BASIC utasításokat.

Jel	*Felhasználás*	*Jel*	*Felhasználás*
"" (idézőjel)	Füzérállandókat zár be.	# (sorszám)	I/O utasításoknál megelőzi a logikai file-számot.
? (kérdőjel)	A PRINT kulcsszó rövid alakja.	$ (dollár)	Füzérváltozót jelöl, gépi nyelvű monitornál hexadecimális szám jelölésére szolgál.
((bal zárójel)	Kifejezés értékelése és függvények.	& (és)	Gépi nyelvű monitornál oktális számot jelöl.
) (jobb zárójel)	Kifejezés értékelése és függvények.	π (pi)	A 3.141592654 numerikus állandó jelölésére szolgál.
% (százalék)	Változóneveket egésznek nyilvánít, gépi nyelvű monitornál bináris számot jelöl.		

FÜGGELÉK

A) Függelék. A BASIC nyelv hibaüzenetei 196

B) Függelék. A DOS hibaüzenetei 199

C) Függelék. Kapcsolók és csatlakozók a külső egységekhez 202

D) Függelék. Képernyőkódok 205

E) Függelék. Az ASCII és a CHR$ kódok 207

F) Függelék. A képernyő és a színtár térképe 209

G) Függelék. Származtatott matematikai függvények 211

H) Függelék. Tártérkép 212

I) Függelék. Vezérlő- és ESCAPE kódok 215

J) Függelék. Gépi kódú monitor 218

K) Függelék. A BASIC 7.0 rövidítései 224

L) Függelék. A lemez parancsok összefoglalása 226

M) Függelék. Szakkifejezések magyarázata 227

A BASIC NYELV HIBAÜZENETEI

Az itt következő hibaüzenetek a BASIC nyelv részei. Hibaüzeneteket az ERR$() függvénnyel is lehet jelezni. A következő táblázatban megadott hibaszámok csak azokra a számokra vonatkoznak, amelyek az ERR$() függvénnyel használva vannak fenntartva.

A hiba száma	A hiba neve	A hiba leírása
1	TOO MANY FILES	Egyidejűleg csak 10 file nyitható meg OPEN utasítással.
2	FILE OPEN	Kísérlet egy már megnyitott file megnyitására.
3	FILE NOT OPEN	Az I/O utasításban megadott számú file-t előzőleg meg kell nyitni.
4	FILE NOT FOUND	Vagy nincs ilyen nevű file (a lemezen), vagy az EOT (end of tape = szalag vége) jelzés elolvasása megtörtént.
5	DEVICE NOT PRESENT	A kért I/O egység nem áll rendelkezésre vagy (kazettás egységnél) a pufferek nincsenek a helyükön. Meg kell vizsgálni, hogy a kért egység csatlakoztatva van-e és be van-e kapcsolva.
6	NOT INPUT FILE	A program olyan file-ból akart GET vagy INPUT utasításokkal adatokat beolvasni, amelyet kivitelként jelöltünk ki.
7	NOT OUTPUT FILE	A program olyan file-ba akar adatokat írni, amelyet bevitelként jelöltünk ki.
8	MISSING FILE NAME	A file neve hiányzik az utasításból.
9	ILLEGAL DEVICE NUMBER	A program az illető eszközt helytelenül szólította meg (pl. SAVE a képernyőre stb.).
10	NEXT WITHOUT FOR	Vagy a ciklusok egymásbaágyazása helytelen, vagy a NEXT utasítás nem tartalmaz olyan ciklusváltozót, ami megfelelne a FOR utasításban levőnek.
11	SYNTAX	Olyan utasítás, amit a BASIC nem ismer. Okozhatja egy hiányzó vagy a szükségesnél több zárójel, egy véletlenül hibásan begépelt utasításszó stb.
12	RETURN WITHOUT GOSUB	Egy RETURN utasítást adtunk ki a hozzá tartozó GOSUB utasítás kiadása előtt.
13	OUT OF DATA	A READ utasítás még akkor is olvasni akar, amikor a DATA sorokból már elfogytak az adatok.
14	ILLEGAL QUANTITY	Egy függvény vagy utasítás argumentumaként megadott szám a megengedett határokon kívülre esik.
15	OVERFLOW	A számítás eredménye nagyobb, mint a megengedett legnagyobb szám, amely 1.701411834E + 38.

A hiba száma	A hiba neve	A hiba leírása
16	OUT OF MEMORY	Vagy a programkód és/vagy a program változói számára nincs elég hely, vagy túl sok az egymásba ágyazott DO, FOR vagy GOSUB utasítás.
17	UNDEF'D STATEMENT	A program egy nemlétező sorszámra hivatkozott.
18	BAD SUBSCRIPT	A program egy olyan tömbelemre hivatkozik, ami a DIM utasításban megadott tömbhatáron kívülre esik.
19	REDIM'D ARRAY	Egy tömb mérete csak egyszer határozható meg.
20	DIVISION BY ZERO	Osztás nullával (nem megengedett).
21	ILLEGAL DIRECT	Az INPUT vagy a GET, vagy az INPUT# vagy a GET# utasítások csak programon belül használhatók.
22	TYPE MISMATCH	Ezt a hibajelzést akkor kapjuk, ha szám helyett füzért, ill. füzér helyett számot írtunk.
23	STRING TOO LONG	Egy füzér maximum 255 karaktert tartalmazhat.
24	FILE DATA	A szalagon vagy a lemezen levő file-ból hibás adat került beolvasásra.
25	FORMULA TOO COMPLEX	A számítógép nem értette meg a képletet. A képletet két részre kell bontani vagy kevesebb zárójelet kell használni.
26	CAN'T CONTINUE	A CONT paranccsal nem tudjuk a program futását folytatni, mert vagy nem indítottuk még el a RUN paranccsal, vagy hiba volt a programban, vagy valamelyik sort átszerkesztettük.
27	UNDEF'D FUNCTION	Olyan, a használó által meghatározható függvényre hivatkozunk, amit még nem határoztunk meg.
28	VERIFY	A szalagon, ill. a lemezen tárolt program nem egyezik meg a tárban levő programmal.
29	LOAD	Nem volt hibamentes a betöltés. Meg kell ismételni.
30	BREAK	A program futása közben lenyomtuk a STOP gombot.
31	CAN'T RESUME	A program egy olyan RESUME utasítást tartalmaz, amelyet nem előzött meg TRAP utasítás.
32	LOOP NOT FOUND	A program egy olyan DO utasítást talál, amihez nem találja a hozzá tartozó LOOP utasítást.
33	LOOP WITHOUT DO	A program egy olyan LOOP utasítást talál, amihez nem találja a hozzá tartozó DO utasítást.
34	DIRECT MODE ONLY	A parancs csak direkt módban adható ki, programból nem.
35	NO GRAPHICS AREA	A program valamilyen grafika készítésére vonatkozó utasítást (DRAW, BOX stb.) tartalmaz, még mielőtt a GRAPHIC utasítást végrehajtotta volna.

A hiba száma	A hiba neve	A hiba leírása
36	BAD DISK	Vagy egy még nem formált lemezt akartunk a rövidített (ID nélküli) HEADER utasítással formálni, vagy rossz a lemez.
37	BEND NOT FOUND	A program egy "IF...THEN BEGIN" vagy egy "IF...THEN ...ELSE BEGIN" szerkezetet tartalmaz, és nem talált egy BEND kulcsszót, ami a BEGIN-hez tartozna.
38	LINE# TOO LARGE	Hiba következett be a BASIC program átszámozása közben. A megadott paraméterek egy 63 999-nél nagyobb sorszámot adnak, ezért az átszámozás nem történt meg.
39	UNRESOLVED REFERENCES	A BASIC program átszámozása során hiba történt. Egy utasítás egy olyan sorszámra hivatkozott (pl. GOTO 999), amilyen sorszám nem létezik. Ezért a program átszámozása nem történt meg.
40	UNIMPLEMENTED COMMAND	Olyan parancsot adtunk ki, amit a BASIC 7.0 nem ismer.
41	FILE READ	A gép a lemezmeghajtóról egy program vagy file betöltése vagy beolvasása közben hibát észlel (pl. egy program betöltése közben kinyitottuk a meghajtó ajtaját).

A DOS HIBAÜZENETEI

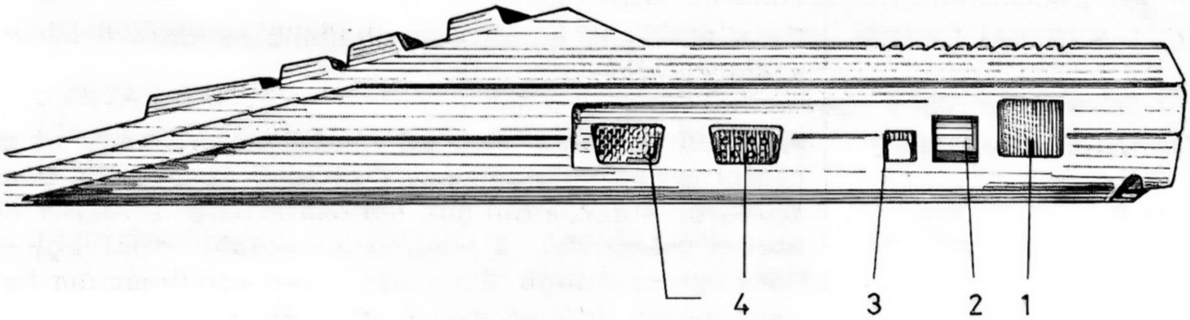
Az itt következő DOS hibaüzeneteket a DS és a DS$ változók tartalmazzák. A DS változó csak a hiba számát, a DS$ változó a hiba számát, a hibaüzenetet és az illető sáv és szektor számát tartalmazza. Figyelem: a 20-nál kisebb számú hibaüzeneteket figyelmen kívül kell hagyni a 01-es kivételével, ami a SCRATCH utasítással törölt file-ok számát tartalmazza.

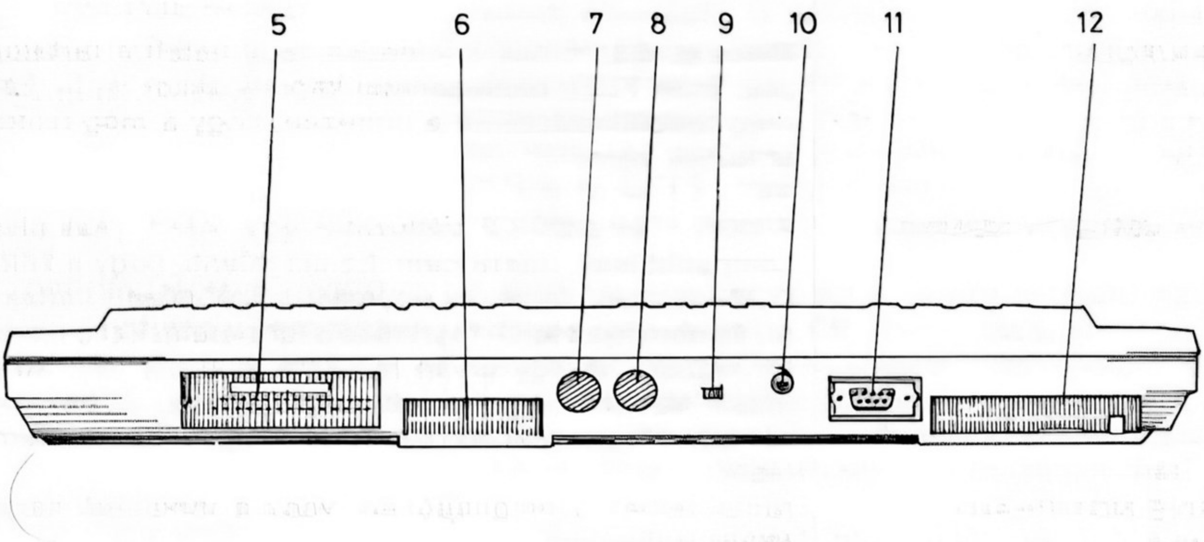
A hiba száma	A hiba neve	A hiba leírása
20	READ ERROR (nincs meg a blokk fejléce)	A lemezvezérlő nem tudja behatárolni a keresett adatblokk fejlécét. A hibát egy illegális szektorszám vagy az adatblokk sérült fejléce okozhatja.
21	READ ERROR (nincs szinkronjel)	A lemezvezérlő a kívánt sávban nem képes szinkronjelet érzékelni. Okozhatja az író/olvasófej hibája, vagy nincs lemez, vagy a lemez nem formált vagy rosszul van behelyezve. Hardverhibát is jelezhet.
22	READ ERROR (nincs adatblokk)	A lemezvezérlőnek egy helytelenül felírt adatblokkot kell elolvasni vagy ellenőrizni. Ez a hiba a BLOCK utasításokkal kapcsolatban jelentkezik, és azt jelzi, hogy a kérés vagy illegális sávra és/vagy szektorra irányult.
23	READ ERROR (ellenőrző összeg hiba az adatblokkban)	A hibaüzenet azt jelzi, hogy egy vagy több adatbyte hibás. A DOS az adatot beolvassa a tárába, de az ellenőrző összeg (checksum) hibás. Jelenthet hardver földelési problémákat is.
24	READ ERROR (byte dekódolási hiba)	A DOS az adatot vagy a fejlécet beolvasta a tárába, de az adatbyte-ban levő illegális kód következtében hardverhiba lépett fel. Jelenthet hardver földelési problémákat is.
25	WRITE ERROR (írás-ellenőrzési hiba)	Ezt az üzenetet akkor kapjuk, ha a vezérlő azt állapítja meg, hogy a felírt adat és a DOS tárában levő adat nem azonos.
26	WRITE PROTECT ON (írásvédelem)	Ezt az üzenetet akkor kapjuk, ha a vezérlőnek egy blokkot kellene felírnia, miközben az írásvédelem gomb be van nyomva. Ez a hibajelzés látható akkor is, ha a lemezen az írásvédelemre szolgáló kivágás le van ragasztva.
27	READ ERROR	A kívánt adatblokk fejlécében ellenőrző összeg hiba van. A DOS a blokkot nem olvasta be a tárába.
28	WRITE ERROR	Ezt a hibaüzenetet az váltja ki, hogy egy adatblokk túl hosszú, és felülírja a következő fejléc szinkronjelét.
29	DISK ID MISMATCH (a lemez ID-je hibás)	A hiba akkor lép fel, ha egy nem inicializált lemezhez kell hozzáférni, vagy ha rossz a lemez fejléce.
30	SYNTAX ERROR (általános szintaktikai hiba)	A DOS nem tudja értelmezni a parancscsatornán küldött utasítást. Tipikusan ezt a hibát okozza a file-nevek meg nem engedett száma, pl. ha a COPY parancs bal oldalán két file-név jelenik meg.

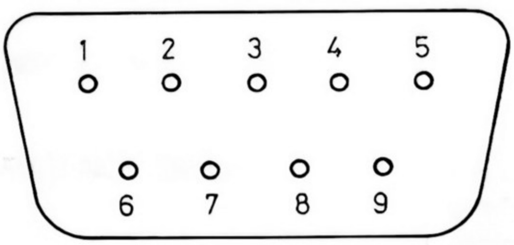
A hiba száma	A hiba neve	A hiba leírása
31	SYNTAX ERROR (nemlétező parancs)	A DOS nem ismeri fel a parancsot. A parancsnak az első pozíción kell kezdődnie.
32	SYNTAX ERROR (nemlétező parancs)	A kiadott parancs hosszabb, mint 58 karakter. A parancsokat rövidített formában kell kiadni.
33	SYNTAX ERROR (nemlétező file-név)	A file neve az OPEN vagy a SAVE parancsban hibásan van megadva. A file nevét ki kell írni.
34	SYNTAX ERROR (nincs megadva file)	Kimaradt a file neve a parancsból vagy a DOS nem tudta file-névként értelmezni. Tipikus hiba, hogy a parancsból kimaradt a kettőspont.
39	SYNTAX ERROR (nemlétező parancs)	Ez a hiba akkor léphet fel, ha a parancscsatornán (a másodlagos cím 15) küldött parancsot a DOS nem ismeri fel.
50	RECORD NOT PRESENT (nincs több rekord)	A DOS az INPUT# vagy a GET# utasításokkal az utolsó rekord olvasásán is túljutott. Ezt a hibaüzenetet kaphatjuk akkor is, ha egy relatív file-nál a pozicionálás a file vége után rekordra irányul. Ha a file-t az új rekorddal (a PRINT# utasítást használva) ki akarjuk bővíteni, akkor a hibaüzenettel nem kell törődni. Ezen hibaüzenetet követően az INPUT# és GET# utasítást nem szabad kiadni anélkül, hogy előzőleg nem végezzük el az újrapozicionálást.
51	OVERFLOW IN RECORD (túlcsordulás a rekordban)	A PRINT# utasítás átlépi a rekord határait. Az információ hiányos lesz. Mivel a CR (kocsivissza) karakter, ami a rekord vége jelként kerül kiadásra, még a rekord részét képezi, ez a hibaüzenet jelenik meg, ha a rekordban a karakterek összes száma (beleértve a rekordot lezáró kocsivissza – CR – karaktert is) túllépi a rekord definiált méretét.
52	FILE TO LARGE	Egy relatív file-on belül olyan nagy rekordszámot használtunk, ami a lemez túlcsordulásához vezetne.
60	WRITE FILE OPEN	Ezt a hibaüzenetet kapjuk, ha egy írásra megnyitott és még nem lezárt file-t ismét meg akarunk nyitni olvasásra.
61	FILE NOT OPEN	Ezt a hibaüzenetet akkor kapjuk, ha egy file-t akarunk elérni, amit a DOS nem nyitott meg. Előfordulhat, hogy nem érkezik hibajelzés; ilyenkor a file elérésére vonatkozó kérés egyszerűen figyelmen kívül marad.
62	FILE NOT FOUND	A keresett file nincs a lemezen.
63	FILE EXISTS	Az újonnan létrehozandó file neve már szerepel a lemezen.
64	FILE TYPE MISMATCH	A kívánt file a parancsban megadott file-típussal nem érhető el. Olvassa el újra a file-típusokról szóló fejezetet.
65	NO BLOCK	A Block Allocation (B–A) paranccsal kapcsolatos hibát jelzi. A lefoglalni kívánt szektor már foglalt. A sávra és a szektorra kapott szám a legközelebbi üres blokk adatait adja meg. Ha a sávra kapott szám nulla, akkor nincs üres szektor. Ha a lemez még nem lenne tele, akkor egy kisebb sáv- és szektorszámmal kell próbálkozni.

A hiba száma	A hiba neve	A hiba leírása
66	ILLEGAL TRACK AND SECTOR	A DOS olyan sávot és szektort akart elérni, ami a megadott párosításban nem létezik. Ez problémát okozhat a következő blokk mutatójának olvasásakor.
67	ILLEGAL SYSTEM T OR S	Ez a speciális hibaüzenet illegális rendszerű sávot vagy szektort jelez.
70	NO CHANNEL (available)	A kívánt csatorna nem áll rendelkezésre vagy az összes csatorna használatban van. Összesen öt csatorna áll rendelkezésre. A soros file-nak két csatornára, a relatív file-nak három csatornára, a hiba/parancscsatornának egy csatornára van szüksége. Ezek bármilyen kombinációja használható, de egy időben maximum csak öt.
71	DIRECTORY ERROR	A BAM (Block Availability Map) nem egyezik meg a DOS tárában levővel. A hiba megszüntetéséhez a lemezt inicializálni kell.
72	DISK FULL	Nincs szabad blokk a lemezen vagy betelt a tartalomjegyzék. DISK FULL hibaüzenetet kapunk akkor is, ha két blokk még rendelkezésre áll a lemezen, hogy a még nyitott file-t le tudjuk zárni.
73	DOS VERSION NUMBER	A DOS 1. és a DOS 2. változatok egymással csak olvasásra kompatíbilisek, írásra nem. Ez azt jelenti, hogy a különböző DOS-változatú lemezek egymást kölcsönösen tudják olvasni, de nem írják felül, mert más a formátum. Ezt a hibaüzenetet kapjuk, ha egy olyan lemezre akarunk írni, amelyik a másik változat szerint lett megformálva. A bekapcsolást követően ugyanezt az üzenetet kapjuk, de ez nem jelez hibát.
74	DRIVE NOT READY	Nincs lemez a meghajtóban, vagy a meghajtó karja vagy ajtaja nyitva van.

KAPCSOLÓK ÉS CSATLAKOZÓK A KÜLSŐ EGYSÉGEKHEZ







1-es számú vezérlőkapu

Láb	*Funkció*	*Megjegyzés*
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	BUTTON A/LP	
7	+5 V	Max. 50 mA
8	GND	
9	POT AX	

2-es számú vezérlőkapu

Láb	*Funkció*	*Megjegyzés*
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	BUTTON B	
7	+5 V	Max. 50 mA
8	GND	
9	POT BX	

A gép jobb oldalán levő csatlakozók

1. A tápegység csatlakozóaljzata – ide csatlakozik a tápegység.
2. Az áramellátás kapcsolója – ezzel lehet a gépet bekapcsolni.
3. RESET gomb – megnyomásával a gépet a kiinduló állapotába állítjuk vissza (melegindítás).
4. Vezérlőkapuk (portok) – két kapu van, egy 1-es és egy 2-es számú. Mindegyik kapuhoz csatlakoztatható botkormány vagy potméter (game controller paddle). Fényceruza csak a gép elejéhez közelebb levő, 1-es számú kapuhoz csatlakoztatható. A kapuk használatára a szoftver ad felvilágosítást.

A gép hátoldalán levő csatlakozók

5. Bővítőkapu – ez a négyszögletes nyílás egy párhuzamos kapu, ahova program- vagy játék-cartridge-ok, valamint speciális illesztőegységek (interface) csatlakoztathatók.

Cartridge bővítő kapu

Láb	Funkció	Láb	Funkció
12	BA	1	GND
13	$\overline{\text{DMA}}$	2	+5 V
14	D7	3	+5 V
15	D6	4	$\overline{\text{IRQ}}$
16	D5	5	R/$\overline{\text{W}}$
17	D4	6	Dot Clock
18	D3	7	I/O 1
19	D2	8	$\overline{\text{GAME}}$
20	D1	9	$\overline{\text{EXROM}}$
21	D0	10	I/O 2
22	GND	11	$\overline{\text{ROML}}$

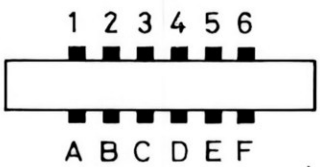
Láb	Funkció	Láb	Funkció
N	A9	A	GND
P	A8	B	$\overline{\text{ROMH}}$
R	A7	C	$\overline{\text{RESET}}$
S	A6	D	$\overline{\text{NMI}}$
T	A5	E	S 02
U	A4	F	A15
V	A3	H	A14
W	A2	J	A13
X	A1	K	A12
Y	A0	L	A11
Z	GND	M	A10

6. A kazettás egység csatlakozóaljzata – a programok és adatok tárolására szolgáló, 1530-as Datasette típusú kazettás magnetofonkészülék csatlakozója.



Kazettás egység csatlakozója

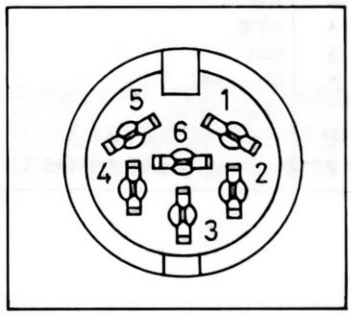
Láb	Funkció
A–1	föld (GND)
B–2	+5 V
C–3	kazettamotor
D–4	kazettaolvasás
E–5	kazettaírás
F–6	billentyűzet



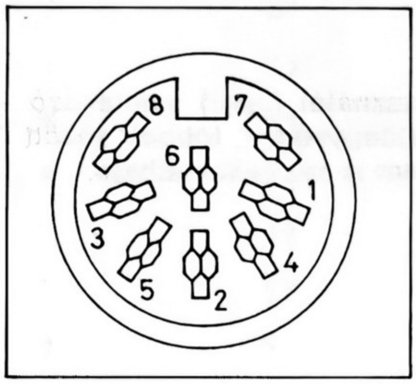
7. Soros kapu – ezen a csatlakozón keresztül a Commodore 128-ashoz közvetlenül kapcsolható egy Commodore soros nyomtató vagy lemezmeghajtó egység.

Soros I/O csatlakozó

Láb	Funkció
1	$\overline{\text{SERIAL}}$ $\overline{\text{SRQIN}}$
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	$\overline{\text{RESET}}$



8. Videocsatlakozó – ezen a DIN csatlakozón közvetlen audio és kompozit videojelek állnak rendelkezésre. Ezek csatlakoztathatók Commodore monitorhoz vagy más egységekhez. Ez a 40 oszlopos kimeneti csatlakozó:



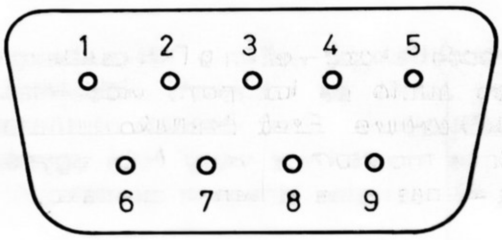
Láb	Funkció	Megjegyzés
1	LUM/SYNC	lumin./SYNC kimenet
2	GND	föld
3	AUDIO OUT	
4	VIDEO OUT	kompozit jel kimenet
5	AUDIO IN	
6	COLOR OUT	kromat. jel kimenet
7	NC	üres
8	NC	üres

9. Csatornaválasztó – ezzel a kapcsolóval kell kiválasztani a megfelelő tv-csatornát (L = 3-as csatorna, H = 4-es csatorna), ha a kép megjelenítésére monitor helyett tv-készüléket használunk.

10. RF csatlakozó – ez a kimenet a televíziós vevőkészülék számára szolgáltat hang- és képjelet. (A tv képernyőjén 40 oszlopos kép jelenik meg.)

11. RGBI csatlakozó – ez a 9 lábú csatlakozó közvetlen audio-jelet és egy RGBI (Red/Green/Blue/Intensity) jelet szolgáltat. Ez a 80 oszlopos kimenet.

Láb	Jel
1	föld
2	föld
3	vörös
4	zöld
5	kék
6	fényerő
7	monokróm
8	vízszintes szinkron
9	függőleges szinkron

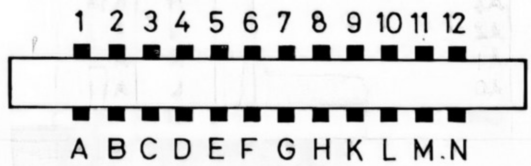


12. Felhasználói (user) csatlakozó – különböző illesztőegységek, többek között Commodore modem is csatlakoztatható.

Felhasználói I/O csatlakozó

Láb	Típus	Megjegyzés
1	GND	
2	+5 V	Max. 100 mA
3	$\overline{\text{RESET}}$	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	$\overline{\text{PC2}}$	
9	SER. ATN IN	
10	9 VAC	Max. 100 mA
11	9 VAC	Max. 100 mA
12	GND	

Láb	Típus	Megjegyzés
A	GND	
B	$\overline{\text{FLAG2}}$	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



KÉPERNYŐKÓDOK

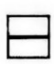
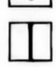
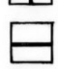
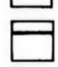
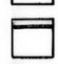
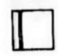
40 oszlopos képernyőkódok

A következő táblázat a Commodore teljes karakterkészletét tartalmazza. Azt is megadja, hogy milyen számot kell POKE utasítással a képernyő tárába (1024–2023 közötti tárcímeken) elhelyezni, hogy a 40 oszlopos képernyőn a kívánt karakter jelenjen meg. (Ne feledkezzünk meg az 55296–56295 közötti tárcímeken található színtár állításáról!). Azt is bemutatja, hogy a képernyőről PEEK utasítással kiolvasott számnak melyik karakter felel meg.

Két karakterkészlet áll rendelkezésre. A 80 oszlopos módban mindkettő egyidejűleg használható, a 40 oszloposban azonban egyszerre csak az egyik. A karakterkészletek átváltása a SHIFT és a C= (Commodore) billentyűk egyidejű lenyomásával történik.

BASIC-ből a nagybetűs/grafikus karakterkészletet a PRINT CHR$(142), a nagybetűs/kisbetűs karakterkészletet a PRINT CHR$(14) utasítás kapcsolja át.

Bármely szám vagy jel INVERZ módon is megjeleníthető. Az inverz karakterek kódját úgy kapjuk, hogy a táblázatban szereplő értékhez hozzáadunk 128-at.

1.készlet	2.készlet	POKE	1.készlet	2.készlet	POKE	1.készlet	2.készlet	POKE
@		0	X	x	24	0		48
A	a	1	Y	y	25	1		49
B	b	2	Z	z	26	2		50
C	c	3	[		27	3		51
D	d	4	£		28	4		52
E	e	5	]		29	5		53
F	f	6	↑		30	6		54
G	g	7	←		31	7		55
H	h	8	SPACE		32	8		56
I	i	9	!		33	9		57
J	j	10	"		34	:		58
K	k	11	#		35	;		59
L	l	12	$		36	<		60
M	m	13	%		37	=		61
N	n	14	&		38	>		62
O	o	15	'		39	?		63
P	p	16	(		40			64
Q	q	17	)		41		A	65
R	r	18	*		42		B	66
S	s	19	+		43		C	67
T	t	20	,		44		D	68
U	u	21	—		45		E	69
V	v	22	.		46		F	70
W	w	23	/		47		G	71

1. készlet	2. készlet	POKE	1. készlet	2. készlet	POKE	1. készlet	2. készlet	POKE
	H	72			91			109
	I	73			92			110
	J	74			93			111
	K	75			94			112
	L	76			95			113
	M	77	SPACE		96			114
	N	78			97			115
	O	79			98			116
	P	80			99			117
	Q	81			100			118
	R	82			101			119
	S	83			102			120
	T	84			103			121
	U	85			104		✓	122
	V	86			105			123
	W	87			106			124
	X	88			107			125
	Y	89			108			126
	Z	90						127

A 128–255 közötti kódok a 0–127 közötti kódok inverz képei

AZ ASCII ÉS A CHR$ KÓDOK

A következő táblázat a PRINT CHR$(X) utasítás hatására megjelenő karaktereket tartalmazza X összes lehetséges értékére. Ugyanígy kikereshető a táblázatból az is, hogy milyen értéket kapunk a PRINT ASC("x") utasítás eredményéül, ahol x bármely megjeleníthető karakter. A táblázat hasznos segítséget nyújt a GET utasítással beolvasott karakter kiértékelésénél, nagybetűs/kisbetűs üzemmódváltásnál és olyan utasítások PRINT-elésénél (mint pl. a karakterkészlet átkapcsoló), amelyek nem szerepelhetnek idézőjelben.

A 192–223 közötti kódok mint 96–127 közötti kódok
A 224–254 közötti kódok mint 160–190 közötti kódok
A 255-ös kód mint 126-os kód

NYOMTAT	CHR$	NYOMTAT	CHR$	NYOMTAT	CHR$	NYOMTAT	CHR$
	0	RED	28	8	56	T	84
	1	CHSR →	29	9	57	U	85
	2	GRN	30	:	58	V	86
	3	BLUE	31	;	59	W	87
	4	SPACE	32	<	60	X	88
WHTE	5	!	33	=	61	Y	89
	6	"	34	>	62	Z	90
	7	#	35	?	63	[	91
letiltja a SHIFT C=	8	$	36	@	64	£	92
engedélyezi a SHIFT C=	9	%	37	A	65	]	93
	10	&	38	B	66	↑	94
	11	'	39	C	67	←	95
	12	(	40	D	68		96
RETURN	13	)	41	E	69		97
SWITCH TO LOWER CASE	14	*	42	F	70		98
	15	+	43	G	71		99
	16	,	44	H	72		100
CHSR	17	–	45	I	73		101
RVS ON	18	.	46	J	74		102
CLR HOME	19	/	47	K	75		103
INST DEL	20	0	48	L	76		104
	21	1	49	M	77		105
	22	2	50	N	78		106
	23	3	51	O	79		107
	24	4	52	P	80		108
	25	5	53	Q	81		109
	26	6	54	R	82		110
	27	7	55	S	83		111

NYOMTAT	CHR$	NYOMTAT	CHR$	NYOMTAT	CHR$	NYOMTAT	CHR$
	112		130	INST DEL	148		166
●	113		131	barna	149		167
	114		132	világospiros	150		168
♥	115	f1	133	sötétszürke	151		169
	116	f3	134	szürke	152		170
	117	f5	135	világoszöld	153		171
╳	118	f7	136	világoskék	154		172
○	119	f2	137	világosszürke	155		173
♣	120	f4	138	PUR	156		174
	121	f6	139	← CRSR	157		175
♦	122		140	YEL	158		176
┼	123	SHIFT RETURN	141	CYN	159		177
	124	SWITCH TO UPPER CASE	142	SPACE	160		178
	125		143		161		179
π	126	BLX	144		162		180
	127	CRSR	145		163		181
	128	RVS OFF	146		164		182
narancs	129	CLR HOME	147		165		183

NYOMTAT	CHR$	NYOMTAT	CHR$	NYOMTAT	CHR$	NYOMTAT	CHR$
	184		186		188		190
	185		187		189		191

A 192 és 223 közötti kódok megegyeznek a 96 és 127 közöttiekkel
A 224 és 254 közötti kódok megegyeznek a 160 és 190 közöttiekkel
A 255-ös kód megegyezik a 126-ossal

F) Függelék

A KÉPERNYŐ ÉS A SZÍNTÁR TÉRKÉPE

A képernyő és a színtár térképe – C128-as üzemmód, 40 oszlop és C64-es üzemmód

A következő térképek azokat a tárcímeket mutatják, amelyeket a 40 oszlopos módban a C128-as és a C64-es a képernyőn levő karakterek és ezek színének az azonosítására használ. A térképek vezérlése egymástól független, és mindegyik 1000 tárcímet tartalmaz.
A térképeken megjelenő karakter közvetlenül vezérelhető a POKE utasítással.

Ha a szín térképére POKE utasítással beírunk egy színkódot, akkor a karakter színe ennek megfelelően változik meg. Pl. a

POKE 55296, 1

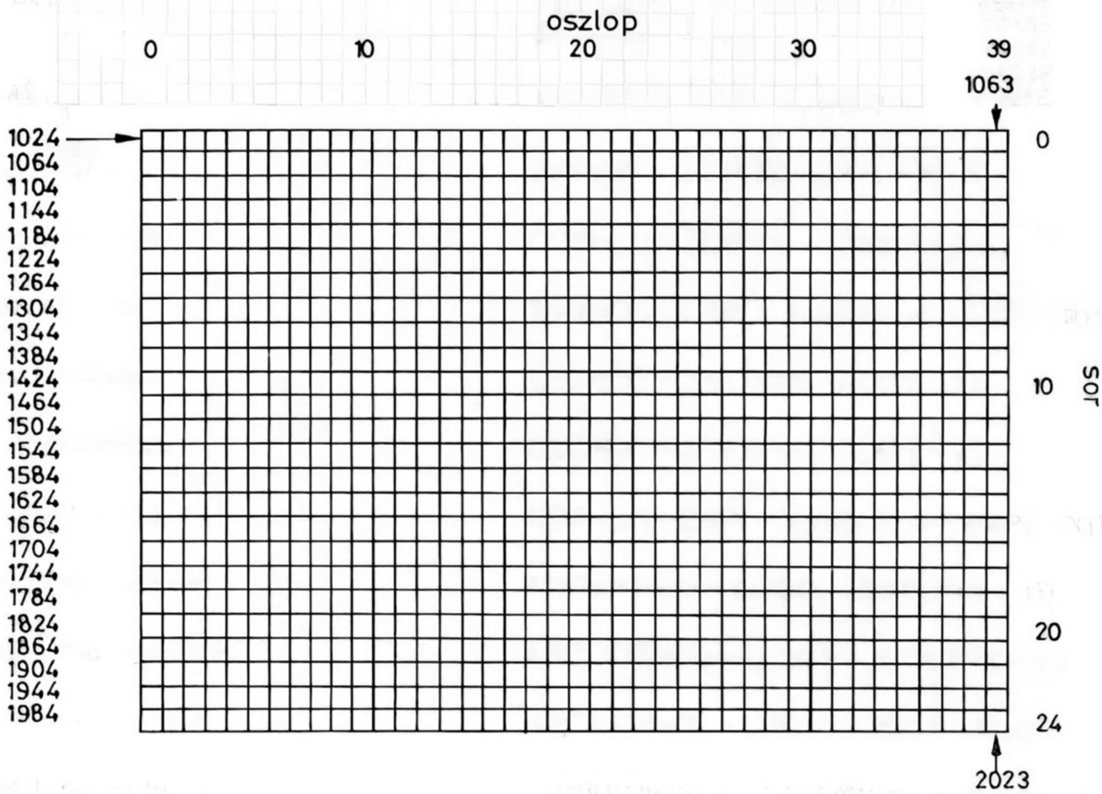
az előbb beírt M betű színét világoskékről fehérre változtatja.

Színkódok – 40 oszlopos ábrázolás

0 fekete
1 fehér
2 piros
3 türkiz

8 narancs
9 barna
10 világospiros
11 sötétszürke

Képernyőtár térképe



A képernyő térképre egy karaktert a POKE utasítással és valamely képernyőkóddal (l. a D) Függelék) írhatunk. Pl. a

POKE 1024, 13

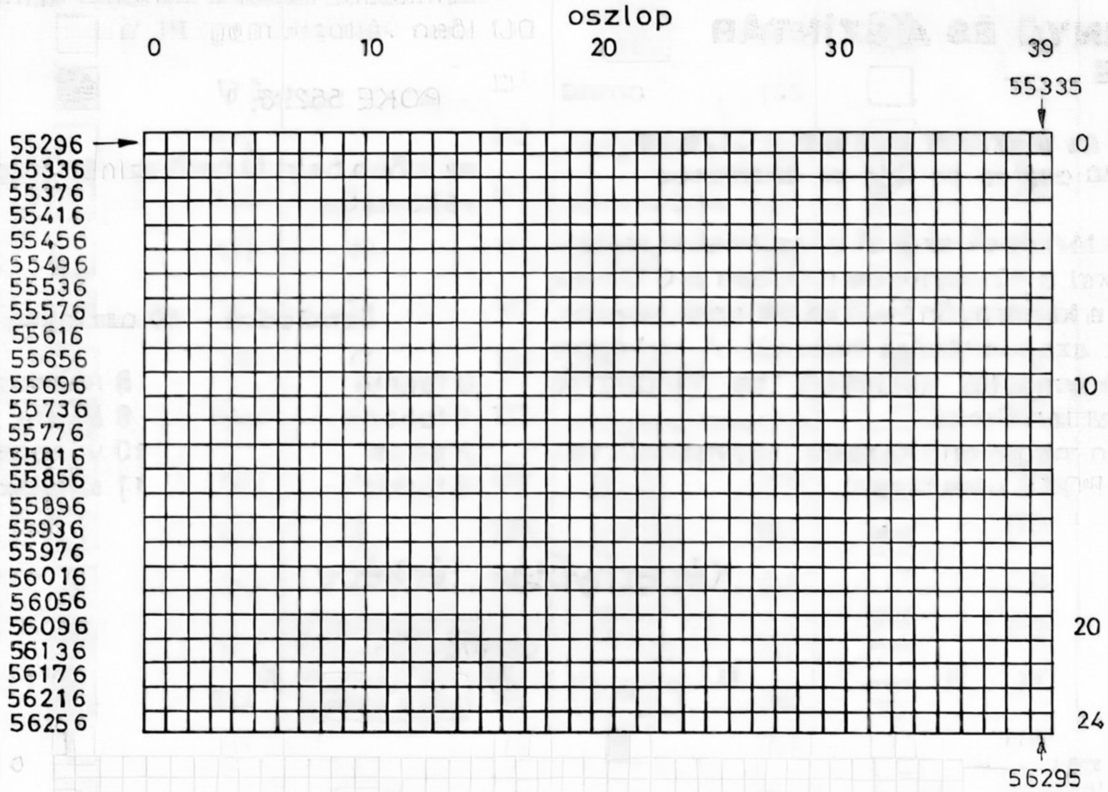
utasítás hatására egy M betű jelenik meg a képernyő bal felső sarkában.

4 bíbor
5 zöld
6 kék
7 sárga

12 középszürke
13 világoszöld
14 világoskék
15 világosszürke

A képernyő keretszínének tárcíme 53 280.
A képernyő háttérszínének tárcíme 53 281.

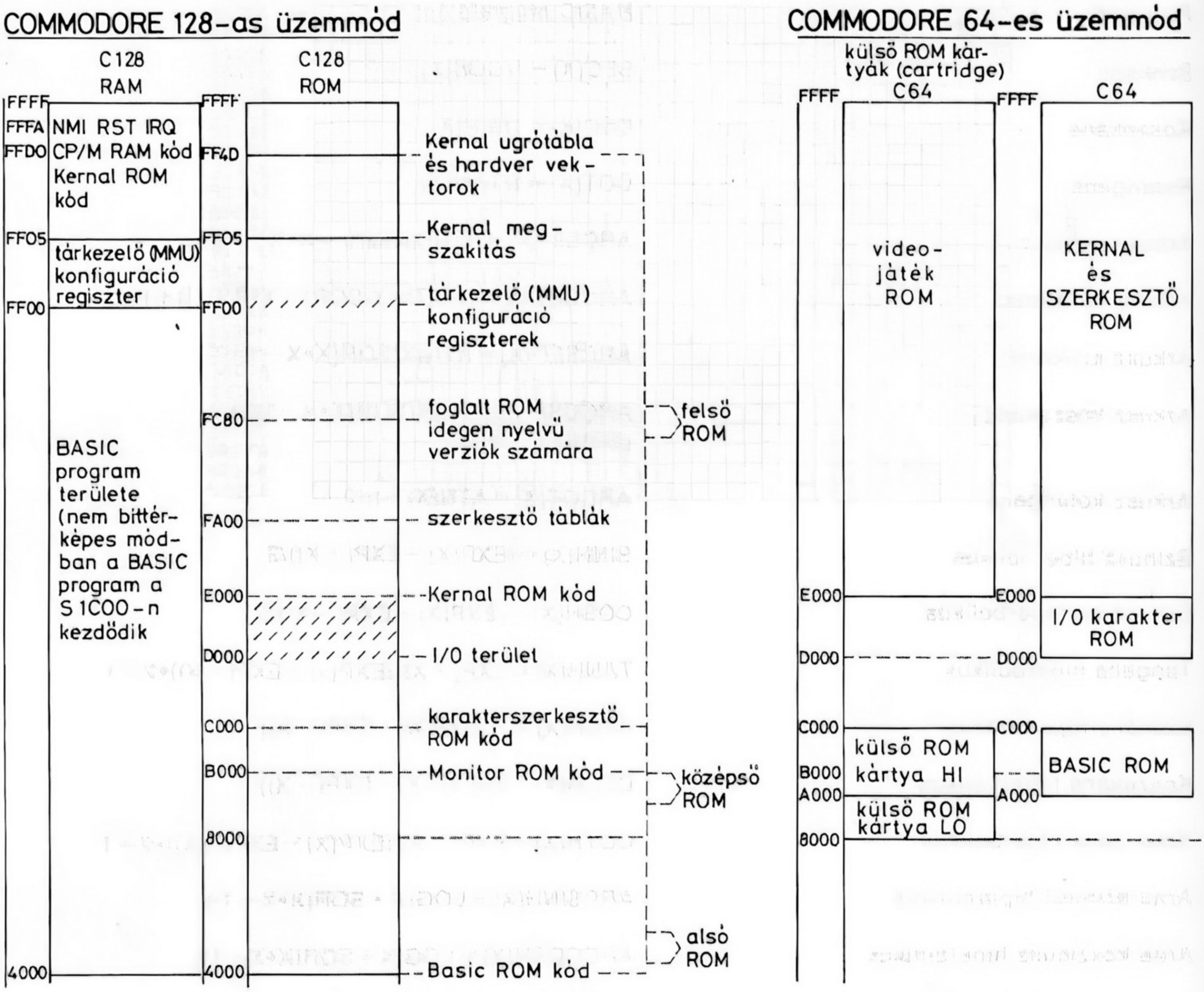
Szintár térképe



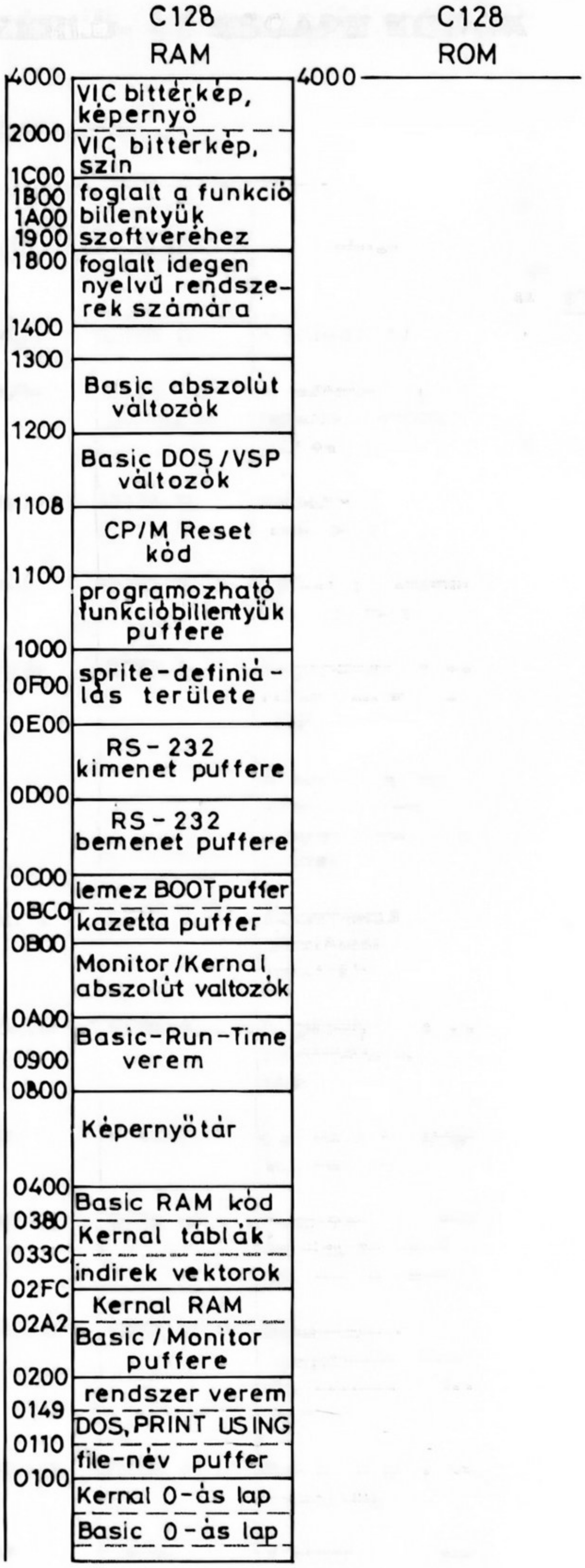
SZÁRMAZTATOTT TRIGONOMETRIKUS FÜGGVÉNYEK

Függvény	*BASIC megfelelője*
Szekáns	SEC(X) = 1/COS(X)
Koszekáns	CSC(X) = 1/SIN(X)
Kotangens	COT(X) = 1/TAN(X)
Arkusz szinusz	ARCSIN(X) = ATN(X/SQR(− X*X + 1))
Arkusz koszinusz	ARCCOS(X) = − ATN(X/SQR(− X*X + 1)) + π/2
Arkusz szekáns	ARCSEC(X) = ATN(X/SQR(X*X − 1))
Arkusz koszekáns	ARCCS(X) = ATN(X/SQR(X*X − 1)) + (SGN(X) − 1*π/2
Arkusz kotangens	ARCOT(X) = ATN(X) + π/2
Szinusz hiperbolikus	SINH(X) = (EXP(X) − EXP(− X))/2
Koszinusz hiperbolikus	COSH(X) = (EXP(X) + EXP(− X))/2
Tangens hiperbolikus	TANH(X) = EXP(− X)/(EXP(x) + EXP(− X))*2 + 1
Szekáns hiperbolikus	SECH(X) = 2/(EXP(X) + EXP(− X))
Koszekáns hiperbolikus	CSCH(X) = 2/(EXP(X) − EXP(− X))
Kotangens hiperbolikus	COTH(X) = EXP(− X)/(EXP(X) − EXP(− X))*2 + 1
Area szinusz hiperbolikus	ARCSINH(X) = LOG(X + SQR(X*X + 1))
Area koszinusz hiperbolikus	ARCCOSH(X) = LOG(X + SQR(X*X − 1))
Area tangens hiperbolikus	ARCTANH(X) = LOG((1 + X)/(1 − X))/2
Area szekáns hiperbolikus	ARCSECH(X) = LOG(SQR((− X*X + 1) + 1/X)
Area koszekáns hiperbolikus	ARCCSCH(X) = LOG((SGN(X)*SQR(X*X + 1/x)
Area kotangens hiperbolikus	ARCCOTH(X) = LOG((X + 1)/(x − 1))/2

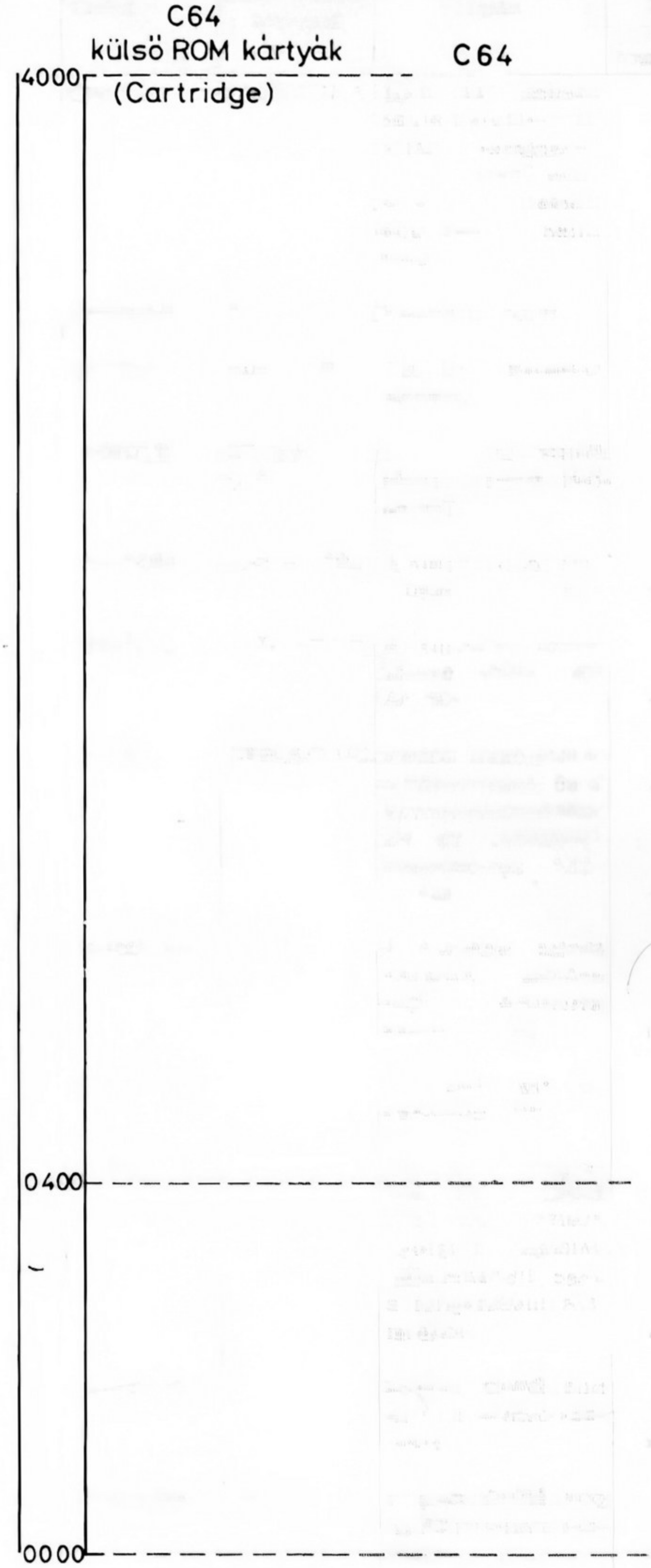
TÁRTÉRKÉP


COMMODORE 128-as üzemmód
C128 RAM
C128 ROM
FFFF
FFFA
NMI RST IRQ
FFD0
CP/M RAM kód
Kernal ROM kód
FF4D
Kernal ugrótábla és hardver vektorok
FF05
tárkezelő (MMU) konfiguráció regiszter
Kernal megszakítás
FF00
tárkezelő (MMU) konfiguráció regiszterek
FC80
foglalt ROM idegen nyelvű verziók számára
felső ROM
BASIC program területe (nem bittérképes módban a BASIC program a $1C00-n kezdődik
FA00
szerkesztő táblák
E000
Kernal ROM kód
D000
I/O terület
C000
karakterszerkesztő ROM kód
B000
Monitor ROM kód
középső ROM
8000
alsó ROM
4000
Basic ROM kód
COMMODORE 64-es üzemmód
külső ROM kártyák (cartridge)
C64
FFFF
video-játék ROM
KERNAL és SZERKESZTŐ ROM
E000
I/O karakter ROM
D000
C000
külső ROM kártya HI
BASIC ROM
B000
A000
külső ROM kártya LO
8000

COMMODORE 128-as üzemmód



COMMODORE 64-es üzemmód



VEZÉRLŐ- ÉS ESCAPE KÓDOK

Vezérlőkódok

CHR$	Billentyűzési sorrend	Hatás	C64	C128
			típusú gépnél használható	
CHR$(2)	CTRL B	Aláhúzás (80)	x	x
CHR$(5)	CTRL 2/CTRL E	A karakter színét fehérre változtatja (40) és (80)	x	x
CHR$(7)	CTRL G	Akusztikus jelet (bell) ad ki		x
CHR$(8)	CTRL H	Letiltja a karakterkészlet cseréjét	x	
CHR$(9)	CTRL I	Engedélyezi a karakterkészlet cseréjét	x	
		A kurzort a legközelebbi beállított tabulátor pozíciójára viszi		x
CHR$(10)	CTRL J	Kocsivissza soremeléssel	x	
		soremelés		x
CHR$(11)	CTRL K	Engedélyezi a karakterkészlet cseréjét		x
CHR$(12)	CTRL L	Letiltja a karakterkészlet cseréjét		x
CHR$(13)	CTRL M	Kocsivissza, soremelés és egy BASIC sor bevitele	x	x
CHR$(14)	CTRL N	Bekapcsolja a nagybetűs/kisbetűs karakter készletet	x	x
CHR$(15)	CTRL O	Bekapcsolja a villogást (80)		x
CHR$(17)	CRSR DOWN/CTRL Q	A kurzort egy sorral lefelé mozgatja	x	x
CHR$(18)	CTRL 9	A karakterek inverze jelenik meg	x	x
CHR$(19)	HOME/CTRL S	A kurzor a képernyő (ill. az aktuális ablak) bal felső sarkába ugrik	x	x

Megjegyzés: (40)... csak 40 oszlopos képernyő esetén
(80)... csak 80 oszlopos képernyő esetén

CHR$	Billentyűzési sorrend	Hatás	C64	C128
			típusú gépnél használható	
CHR$(20)	DEL/CTRL T	Törli az utoljára bevitt karaktert, és a törölt karaktertől jobbra levő valamennyi karaktert egy hellyel balra tolja	x	x
CHR$(24)	CTRL X	Tabulátor be/ki		x
CHR$(27)	ESC/CTRL [	Egy ESC karakter küldése		x
CHR$(28)	CTRL3/ CTRL/	A karakter színét pirosra állítja (40) és (80)	x	x
CHR$(29)	CRSR/CTRL]	A kurzort egy oszloppal jobbra állítja	x	x
CHR$(30)	CTRL6/CTRL	A karakter színét zöldre állítja (40) és (80)	x	x
CHR$(34)	CTRL6/CTRL	Kettős idézőjelet ír a képernyőre, és a képernyőszerkesztőt az „idézőjel" üzemmódba kapcsolja	x	x
CHR$(129)	C= 1	A karakter színét narancs színűre (40); sötétbíborra kapcsolja (80)	x	x
CHR$(130)		Az aláhúzást kikapcsolja (80)		x
CHR$(131)		Elindítja a programot. Ez a CHR$ kód mint PRINT CHR$(131) utasítás nem működik, csak a billentyűzet pufferéből	x	x
CHR$(133)	F1	Foglalt CHR$ kód az F1 billentyű számára	x	x
CHR$(134)	F3	Foglalt CHR$ kód az F3 billentyű számára	x	x
CHR$(135)	F5	Foglalt CHR$ kód az F5 billentyű számára	x	x
CHR$(136)	F7	Foglalt CHR$ kód az F7 billentyű számára	x	x
CHR$(137)	F2	Foglalt CHR$ kód az F2 billentyű számára	x	x

CHR$	Billentyűzési sorrend	Hatás	C64	C128
			típusú gépnél használható	
CHR$(138)	F4	Foglalt CHR$ kód az F4 billentyű számára	x	x
CHR$(139)	F6	Foglalt CHR$ kód az F6 billentyű számára	x	x
CHR$(140)	F8	Foglalt CHR$ kód az F8 billentyű számára	x	x
CHR$(141)	SHIFT RETURN	Kocsivissza és soremelés BASIC sor bevitele nélkül	x	x
CHR$(142)		Bekapcsolja a nagybetűs/grafikus karakterkészletet	x	x
CHR$(143)		Kikapcsolja a villogást (80)		x
CHR$(144)	CTRL 1	A karakter színét feketére állítja (40) és (80)	x	x
CHR$(145)	CRSR UP	A kurzort egy sorral feljebb állítja	x	x
CHR$(146)	CTRL 0	Az inverz megjelenítést kikapcsolja	x	x
CHR$(147)	HOME	Törli a képernyőt és a kurzort a bal felső sarokba állítja	x	x
CHR$(148)	INST	A kurzortól jobbra levő karaktereket egy hellyel jobbra tolja	x	x
CHR$(149)	C= 2	A karakter színét barnára (40); sötétsárgára (80) állítja	x	x
CHR$(150)	C= 3	A karakter színét világospirosra állítja (40) és (80)	x	x
CHR$(151)	C= 4	A karakter színét sötétszürkére (40), sötétcinóberre állítja (80)	x	x
CHR$(152)	C= 5	A karakter színét középszürkére állítja (40) és (80)	x	x
CHR$(153)	C= 6	A karakter színét világoszöldre állítja (40) és (80)	x	x

Megjegyzés: (40)... csak 40 oszlopos képernyő esetén
(80)... csak 80 oszlopos képernyő esetén

CHR$	Billentyűzési sorrend	Hatás	C64	C128
			típusú gépnél használható	
CHR$(154)	C= 7	A karakter színét világoskékre állítja (40) és (80)	x	x
CHR$(155)	C= 8	A karakter színét világosszürkére állítja (40) és (80)	x	x
CHR$(156)	CTRL 5	A karakter színét bíborra állítja (40) és (80)	x	x
CHR$(157)	CRSR LEFT	A kurzort egy oszloppal balra állítja	x	x
CHR$(158)	CTRL 4	A karakter színét türkiz (40); világostürkiz színűre állítja (80)	x	x

Megjegyzés: (40)... csak 40 oszlopos képernyő esetén
(80)... csak 80 oszlopos képernyő esetén

Escape kódok

A következő táblázat a Commodore 128-ason rendelkezésre álló ESCape függvényekhez a bebillentyűzési sorrendet tartalmazza. Ahhoz, hogy egy ESCape műveletet aktiváljunk, először meg kell nyomni az ESC billentyűt, majd elengedni, és ezt követően a táblázatban megadott billentyűt kell megnyomni.

ESCAPE művelet	*ESCAPE billentyű*
Idézőjel és inzert üzemmód törlése	ESC C
Az aktuális sort a kurzor helyétől jobbra törli	ESC Q
Az aktuális sort a sor elejétől a kurzor helyéig törli	ESC P
A kurzor aktuális helyétől a teljes képernyőt törli	EXC @
A kurzor az aktuális sor elejére ugrik	ESC J
A kurzor az aktuális sor utáni első pozícióra ugrik	ESC K
Engedélyezi karakterek automatikus beszúrását (automatikus inzert mód)	ESC A
Letiltja a karakterek automatikus beszúrását	ESC O
Törli az aktuális sort	ESC D

Beszúr egy sort	ESC I
Beállítja az alapértelmezés szerinti tabulálást (8 karakterhely)	ESC Y
Töröl valamennyi tabulálási pontot	ESC Z
Engedélyezi a görgetést	ESC L
Letiltja a görgetést	ECC M
Görgetés egy sorral felfelé	ESC V
Görgetés egy sorral lefelé	ESC W
Engedélyezi az akusztikus jel (bell) kiadását (CTRL = G-vel együtt)	ESC G
Letiltja az akusztikus jel kiadását	ESC H
Megszünteti a kurzor villogását	ESC E
Bekapcsolja a kurzor villogását	ESC F
A kurzor aktuális helye egy képernyő-ablak jobb alsó sarkát definiálja	ESC B
A kurzor aktuális helye egy képernyő-ablak bal felső sarkát definiálja	ESC T
Átkapcsol a 40/80 oszlopos monitor között	ESC X

A következő ESCape műveletek csak a 80 oszlopos monitorra vonatkoznak. (L. a 80 oszlopos képernyőre vonatkozó 8-as alfejezetet.)

Átkapcsol vonalkurzorra	ESC U
Átkapcsol négyzetkurzorra	ESC S
A képernyőt a normál kijelzésről inverz kijelzésre kapcsolja	ESC R
A képernyőt inverz kijelzésről normál kijelzésre kapcsolja	ESC N

J) Függelék

GÉPI KÓDÚ MONITOR

Bevezetés

A Commodore 128-asba be van építve egy gépi kódú monitorprogram, ami a használó számára lehetővé teszi gépi kódú programok írását és vizsgálatát. A Commodore 128 MONITOR egy gépi kódú monitorból, egy mini-assemblerből és egy disassemblerből áll. A beépített monitorprogram csak a C128-as üzemmódban használható, 40 vagy 80 oszlopos képernyőformátumban. A Commodore 128 MONITOR segítségével megírt gépi kódú programok futhatnak önmagukban vagy felhasználhatók BASIC programok igen gyors szubrutinjaiként, mivel a Commodore 128 MONITOR alkalmas arra, hogy a BASIC-kel békésen együttműködjön.

Arra kell csak ügyelni, hogy az assembly nyelvű programokat úgy helyezzük el a tárban, hogy azokat a BASIC program ne írja felül.

BASIC-ből a monitorba való belépéshez gépeljük be:

MONITOR **RETURN**

A Commodore 128 MONITOR parancsai

KULCSSZÓ	FUNKCIÓ	FORMÁTUM
ASSEMBLE	Assemblálja a 8502-es utasításkódját	A <kezdőcím> <műv. kód> [operandus]
COMPARE	Összehasonlít két tárterületet, és kijelzi az eltérést	C <kezdőcím> <végcím> <új kezdőcím>
DISASSEMBLE	Disassemblálja a 8502-es egy vagy több gépi kódú sorát	D [<kezdőcím>] [<végcím>]
FILL	Feltölt egy tárterületet egy megadott byte-tal	F <kezdőcím> <végcím> <byte>
GO	A megadott címtől kezdődően elindítja a program végrehajtását	G [cím]
HUNT	A megadott tárterületen megkeresi egy megadott byte-sorozat összes előfordulási helyét	H <kezdőcím> <végcím> <1. byte> [<n. byte>...] H <kezdőcím> <végcím> '<ASCII karakterlánc>
JUMP	Ugrás szubrutinra	J [cím]
LOAD	Betölt egy file-t szalagról vagy lemezről	L "<file-név>" [,<eszköz#> [,<töltés kezdőcíme>]]
MEMORY	Kijelzi a tárhelyek hexadecimális értékeit	M [<kezdőcím> [végcím]]
REGISTERS	Kijelzi a 8502-es regisztereit	R
SAVE	Mentés szalagra vagy lemezre	S "<file-név>", <eszközszám>, <kezdőcím> <végcím + 1>
TRANSFER	Adatok átvitele a tár egyik részéből a másik részébe	T <kezdőcím> <végcím> <új kezdőcím>
VERIFY	Összehasonlítja a tár tartalmát a szalagon vagy lemezen levő tartalommal	V "<file-név>" [,eszközszám> [,<töltés kezdőcíme>]]
EXIT	Kilépés a Commodore 128 MONITOR üzemmódból	X
.	Assemblálja a 8502-es utasításkódját	.
>	Módosítja a tárat	>
;	Módosítja a regisztereket	;
@	Kijelzi a lemez állapotát, kiad lemezre vonatkozó parancsot, kijelzi a tartalomjegyzéket lemez állapota parancs a lemezhez lemez tartalomjegyzék	@ @ [eszközszám] @ [eszközszám] [,<parancs-füzér>] @ [eszközszám], $ [[meghajtó][:<file.spec.>]]

MEGJEGYZÉS: a < > közötti paraméterek megadása kötelező
a [ ] közötti paraméterek megadása opcionális

A Commodore 128-as a gépi kódú monitorában 5 helyiértéken jelzi ki a hexadecimális tárcímeket. Normál esetben a rendelkezésre álló tárterületet egy négy helyiértékű hexadecimális szám jelöli. A balról legszélső (legnagyobb helyiértékű) hexadecimális számjegy (az adott utasítás végrehajtásának idején fennálló) bankkonfigurációt mutatja a következő tárkonfigurációknak megfelelően:

0–RAM 0 csak	8–EXT ROM, RAM 0, I/O
1–RAM 1 csak	9–EXT ROM, RAM 1, I/O
2–RAM 2 csak	A–EXT ROM, RAM 2, I/O
3–RAM 3 csak	B–EXT ROM, RAM 3, I/O
4–INT ROM, RAM 0, I/O	C–KERNAL + INT/lo/,RAM 0, I/O
5–INT ROM, RAM 1,I/O	D–KERNAL + EXT/lo/,RAM 1, I/O
6–INT ROM, RAM 2, I/O	E–KERNAL + BASIC,RAM 0, CHARROM
7–INT ROM, RAM 3, I/O	F–KERNAL + BASIC,RAM 0, I/O

A monitormezőket jellemző írásjelek

A következő írásjelek a monitor adatmezői (pl. tártartalom) előtt állnak. Ezek az írásjelek parancs módban kiadva arra utasítják a monitort, hogy a tár vagy a regiszterek tartalmát a megadott értékre változtassa meg.

. A disassemblált sorok előtt áll.
> A tártartalmat kijelző sorok előtt áll.
; A regisztertartalmat kijelző sorok előtt áll.

A következő írásjelek számértékeket (pl. címeket) tartalmazó mezők előtt állnak, és azt határozzák meg, hogy a monitor az így kapott értékeket melyik számrendszerben értelmezze.

üres Alapértelmezés, hexadecimális értékek.
$ Hexadecimális értékek előtt áll (16-os számrendszer).
+ Decimális értékek előtt áll (10-es számrendszer).
& Oktális értékek előtt áll (8-as számrendszer).
% Bináris értékek előtt áll (2-es számrendszer).

A következő karaktereket a monitor az egyes mezők elválasztására vagy a sorok lezárására használja (kivéve, ha ezek nem egy ASCII füzér részeként fordulnak elő).

üres (szóköz) két mezőt választ el.
, Két mezőt választ el.
: Egy sor logikai lezárása.
? Egy sor logikai lezárása.

A Commodore 128-as MONITOR parancsok

A korábbiakban említetteket leszámítva jelenleg nincsenek változások a MONITOR parancsokban. Figyelembe kell azonban venni, hogy valamennyi szám mező (pl. címek, eszközazonosító számok és adatbyte-ok) kiinduló számként specifikálható. Ez az ASSEMBLE parancs operandusmezejére is vonatkozik. Vegye figyelembe a lemezkezelésre vonatkozó parancsok szintaxisának kiegészítését is.

A programozóknak további segítség, hogy a monitor módban való munka közben a Kernal hibaüzenetei továbbra is olvashatók. Ez azt jelenti, hogy ha a MONITORból valamilyen nem megengedett I/O hozzáférést kísérelnénk meg, akkor a Kernal "I/O ERROR #" hibajelzést és a hiba kódját adja ki. A MONITOR-ból való kilépéssel ez a hibajelzési szolgáltatás megszűnik.

PARANCS: **A**
CÉLJA: Egy assembly kódú sor beírása
SZINTAXIS: **A** < cím > <műv. kód mnemonik> <operandus>

<cím> Annak a tárcímnek a száma, ahová a műveleti kódot el kell helyezni.

<műv. kód mnemonik> Standard MOS technológiájú, assembly nyelvű mnemonik, mint pl. LDA, STX, ROR.

<operandus> Az operandus – ha egyáltalán szükséges – valamennyi megengedett címzési módban szerepelhet.

Az assembly utasítássor beírásának befejezését a RETURN billentyű benyomásával jelezzük. Ha a sor beírása közben valahol hibáztunk, akkor a sorban megjelenik egy kérdőjel, és a kurzor a következő sorba ugrik. A hibás sorban a hiba kijavítására a képernyőszerkesztő mód használható.

PÉLDA
.A01200 LDX # $00
.A01201
FIGYELEM: egy (.) írásjel megfelel az ASSEMBLE parancsnak.
PÉLDA:
.02000 LDA #$23

PARANCS: **C**
CÉLJA: a tár két részének összehasonlítása
SZINTAXIS: **C** <kezdőcím> <végcím> <új kezdőcím>

<kezdőcím> Annak a tárterületnek a kezdőcíme, amelyet a másik területtel össze akarunk hasonlítani.
<végcím> Annak a tárterületnek a végcíme, amelyet a másik területtel össze akarunk hasonlítani.
<új kezdőcím> Annak a tárterületnek a kezdőcíme, amivel az előzőekben megadott területet össze akarjuk hasonlítani.

PARANCS: **D**
CÉLJA: a gépi kód disassemblálása assembly nyelvű mnemonikokká és operandusokká.
SZINTAXIS: **D** [<kezdőcím>]
[<végcím>]

<kezdőcím> Ettől a címtől kezdődik a disassemblálás.
<végcím> Az utolsó disassemblálandó kódcím (megadása nem kötelező).

A disassembly formátuma némileg eltér az assembly beírási formátumától. A különbség annyi, hogy egy disassembly sor első karaktere nem A, hanem egy pont (a könnyebb olvashatóság kedvéért), és a kód hexadecimális értéke is listázásra kerül.
A disassembly lista a képernyőszerkesztővel módosítható. A képernyőn felülírhatjuk a menmonikot vagy az operandust, majd a kocsivissza (RETURN) billentyűt lenyomva a monitor ezt mint egy assembler sort veszi tudomásul, és bejelentkezik az assembler a további utasításokért.
A disassembly lista lapozható. Ha lenyomjuk a D billentyűt majd a RETURN-t, akkor a disassembly lista következő oldala kerül a képernyőre.
PÉLDA:
D 3000 3003
.03000 A900 LDA #$00
.03002 FF ???
.03003 D02B BNE $3030

PARANCS: **F**
CÉL: egy tárterület feltöltése a megadott byte-tal.
SZINTAXIS: **F** <kezdőcím> <végcím> <byte>

<kezdőcím> Az első tárcím, amelyre a megadott byte-ot be kell írni.
<végcím> Az utolsó tárcím, amelyre a megadott byte-ot be kell írni.
<byte értéke> A beírandó 1 vagy 2 helyjegyes hexadecimális szám.

Ezt a parancsot hasznosan lehet alkalmazni adatstruktúrák vagy egyéb RAM-területek nullázására.

PÉLDA:
F 0400 0518 EA
Ez a parancs a $0400 és a $0518 közötti tárcímeket $EA-val (üres, NOP utasítás) tölti fel.

PARANCS: **G**
CÉL: elkezdi a program végrehajtását egy megadott címtől
SZINTAXIS: **G** [<cím>]

<cím> Az a cím, amelytől a program végrehajtása elkezdődik. Ha nem adunk meg címet, akkor a végrehajtás a programszámláló (PC) aktuális értékétől kezdődik. (A PC aktuális értékét az R parancssal jelezhetjük ki.)

A GO parancs a program indítása előtt elmenti a regiszterek aktuális tartalmát (kijelezhető az R parancssal), majd a megadott kezdőcímtől megkezdi a program végrehajtását. A GO parancsot óvatosan használjuk! Ahhoz, hogy a gépi kódú program végrehajtása után ismét visszatérhessünk a Commodore 128-as MONITOR-ba, a végrehajtandó programnak egy BRK utasítással kell befejeződnie.

PÉLDA:
G 140C
A program végrehajtása a $140C tárcímtől kezdődik.

PARANCS: **H**

CÉL: Egy meghatározott tárterületen egy byte-sorozat valamennyi előfordulási helyének megkeresése.

SZINTAXIS: **H**	<kezdőcím>	<végcím> <adatok>
	<kezdőcím>	Az a tárcím, amelytől a keresés kezdődik.
	<végcím>	Az a tárcím, ameddig a keresés tart.
	<adatok>	A keresett adatsor, amely lehet hexadecimális vagy ASCII füzér.

PÉLDA:
H A000 A101 A9 FF 4C
a $A9, $FF, $4C byte-sorozat keresése az A000 és az A101 tárcímek közötti területen.
H 2000 9800 'CASH'
a "CASH" füzér keresése.

PARANCS: **L**

CÉL: egy file betöltése szalagról vagy lemezről.

SZINTAXIS: **L**	<"file-név">	[,<eszköz>[,töltés kezdőcíme]]
	<"file-név">	Bármely, a Commodore 128-as által megengedett file-név
	<eszköz>	Annak a készüléknek a jelzőszáma, amelyről a betöltés történik. Az 1-es a kazettás egységet, a 8-as (vagy 9, A stb.) lemezegységet jelent.
	[töltés kezdőcíme]	A file-t a megadott címtől kezdődően tölti be (opcionális).

A LOAD parancs egy file-t tölt a tárba. A kezdőcímet a lemezes file (programfile) első két byte-ja tartalmazza. Más szavakkal, a LOAD parancs a file-t mindig ugyanarra a tárterületre tölti be, ahonnan az kimentésre került. Ez a gépi kódban végzett munka esetén nagyon fontos, mert nagyon kevés programot lehet teljes egészében más tárterületre áthelyezni. A tárba a teljes file töltődik be (EOF-ig).

PÉLDA:
L "PROGRAM",8 betölti a PROGRAM nevű file-t a lemezről.

PARANCS **M**

CÉL: A megadott tárterületen a tár tartalmának kijelzése hexadecimális és ASCII kódban.

SZINTAXIS: **M**	[<kezdőcím>]	[<végcím>]
	<kezdőcím>	A kijelzésre kerülő tárterület kezdőcíme. Opcionális. Ha nem adjuk meg, akkor egy oldal kerül kijelzésre. Az első byte a bank számát, a következő négy byte a kijelzendő terület első tárcímét adja meg.
	<végcím>	A kijelzésre kerülő tárterület utolsó címe. Opcionális. Ha nem adjuk meg, akkor egy oldal kerül kijelzésre. Az első byte a bank számát, a következő négy byte a kijelzendő terület utolsó címét adja meg.

A tár kijelzésének formátuma:
>1A048 41 E7 00 AA AA 00 98 56 45:A!.*..VE

A kijelzett tártartalmat a képernyőszerkesztővel módosítani lehet. Vigye a kurzort a módosítani kívánt adatra, írja be a kívánt módosítást, és nyomja le a RETURN billentyűt. Ha ez a módosítás egy rossz RAM tárcímre vonatkozik, vagy a ROM átírását kísérelné meg, akkor a képernyőn egy kérdőjel (?) jelenik meg. A képernyőn a kijelzett adatok ASCII kódú megfelelője (a képernyőn megjelenő adatokkal ellentétben) INVERZ módban jelenik meg a képernyő jobb oldalán. A nem kijelezhető karaktert egy inverz pont jelöli. A disassembly parancshoz hasonlóan az M és a RETURN billentyűk lenyomásával a tárban lapozni lehet.

PÉLDA:
M 21C00 21C10
>21C00 41 E7 00 AA AA 00 98 56 45:A!.*..VE
>21C08 42 43 02 AZ AD 11 94 57 44:BC.*..WD
>21C10 45 E7 00 DF FE 07 06 46 47:E!.*..EF
Megjegyzés: a fenti kijelzés a 40 oszlopos üzemmódra vonatkozik.

PARANCS: **R**

CÉL: Kijelzi a 6502-es fontos regisztereinek a tartalmát. Kijelzi a programszámláló, az akkumulátor, az X és az Y index regiszterek és a veremmutató tartalmát.

SZINTAXIS: **R**

PÉLDA:

```
R
PC      SR AC XR YR SP
; 01002 01 02 03 04 F6
```

Megjegyzés: a pontosvessző (;) ugyanúgy használható a regiszterek tartalmának változtatására, mint a > jel a tártartalmak változtatására.

PARANCS: **S**

CÉL: A tár tartalmának kimentése szalagra vagy lemezre.

SZINTAXIS: **S** <"file-név"> , <eszköz> , <kezdőcím> , <végcím>

<"file-név">	Bármely, a Commodore 128-as által megengedett file-név. Az adatok kimentéséhez a file nevét kettős idézőjelbe kell tenni. Az egyszeres idézőjel nem használható.
<eszköz>	Ennek a száma azt jelzi, hogy a file-t hová kell kimenteni. A kazetta száma 01; a lemezegység száma 08, 09 stb. lehet.
<kezdőcím>	A kimentendő tártartalom kezdőcíme.
<végcím>	A kimentendő tártartalom végcíme + 1. Az utolsó címen levő adatot kivéve valamennyi adat kimentésre kerül.

Az ezzel a paranccsal létrehozott file egy programfile. A file első két byte-ja a file kezdőcímét tartalmazza. A file-t az L paranccsal újra be lehet tölteni.

PÉLDA:

```
S "GAME",8,0400,0BFF
```

A parancs a $0400–$0BFF közötti tártartalmat lemezre menti.

PARANCS: **T**

CÉL: Egy tárterület tartalmának átvitele egy másik tárterületre.

SZINTAXIS: **T** <kezdőcím> <végcím> <új kezdőcím>

<kezdőcím>	Az átviendő tárterület kezdőcíme.
<végcím>	Az átviendő tárterület végcíme.
<új kezdőcím>	Annak a tárterületnek a kezdőcíme, ahová az adatokat át akarjuk vinni.

Az adatok az alacsonyabb tárcímekről a magasabb tárcímekre és fordítva is átvihetők. Ehhez kapcsolódó, bármely hosszúságú tárterület is mozgatható mindkét irányban. Az egyes byte-ok átvitelekor végrehajtódik egy automatikus összehasonlítás („compare"), és az eltérések címek szerint listázásra kerülnek.

PÉLDA:

```
T 1400 1600 1401
```

Az adatokat a $1400-as tárcímtől a $1600-as tárcímig terjedő területről a kezdőcímhez képest egy byte-tal magasabb kezdőcímű területre viszi.

PARANCS: **V**

CÉL: Egy szalagon vagy lemezen levő file-t összehasonlítja a tárban levő file-lal.

SZINTAXIS: **V** <"file-név"> [, eszköz][, kezdőcím]

<"file-név">	A Commodore 128-as bármely érvényes file-neve.
<eszköz>	Annak az eszköznek a száma, amelyen a file található; a kazetta száma 01, a lemezegység száma 08, 09 stb.
<kezdőcím>	Opció az ezen a címen kezdődő összehasonlításhoz.

A VERIFY parancs összehasonlítja egy file tartalmát a tár tartalmával. A Commodore 128-as VERIFYING-gal válaszol. Ha az összzehasonlítás során hibát észlel, akkor a VERIFYING ERROR üzenetet küldi. Ha az összehasonlítás sikeres volt, akkor a kurzor megjelenik.

PÉLDA:

```
V"WORKLOAD",08
```

PARANCS: **X**

CÉL: Kimenet a BASIC-hez

SZINTAXIS: **X**

PARANCS: > (nagyobb mint)
CÉL: Segítségével 1–8 tárcím tartalma írható felül egyidejűleg.
SZINTAXIS: ><cím> <1.adatbyte> <2...8 adatbyte>

<cím>	Első beállítandó tárcím.
<1. adatbyte>	A címre írandó adat.
<2...8 adatbyte>	Az azt követő tárcímekre beírandó adatok, egymástól egy-egy szóközzel elválasztva.

PARANCS: @
CÉL: A lemez állapotát jelzi ki.
SZINTAXIS: @ [<egységszám>], <lemez utasítás füzér>

<egységszám>	Az egység száma (opcionális).
<lemezre vonatkozó füzérutasítás>	

Megjegyzés: @ Önmagában a lemezmeghajtó állapotát mutatja.
PÉLDÁK:
@ Lemez állapot vizsgálata
00, 0K, 00, 00
@,I A 8-as meghajtóegységet inicializálja

K) Függelék

A BASIC 7.0 RÖVIDÍTÉSEI

Megjegyzés: A következő rövidítések a nagybetűs grafikus üzemmódban használhatók. Be kell billentyűzni a következőkben felsorolt betűket, majd a SHIFT billentyűt lenyomva tartva be kell billentyűzni a SHIFT-et követő betűt.

Kulcsszó	*Rövidítés*
ABS	A SHIFT B
APPEND	A SHIFT P
ASC	A SHIFT S
ATN	A SHIFT T
AUTO	A SHIFT U
BACKUP	BA SHIFT C
BANK	B SHIFT A
BEGIN	B SHIFT E
BEND	BE SHIFT N
BLOAD	B SHIFT L
BOOT	B SHIFT O
BOX	nincs
BSAVE	B SHIFT S
BUMP	B SHIFT U
CATALOG	C SHIFT A
CHAR	CH SHIFT A
CHR$	C SHIFT H
CIRCLE	C SHIFT I
CLOSE	CL SHIFT O
CLR	C SHIFT L
CMD	C SHIFT M
COLLECT	COLL SHIFT E
COLINT	nincs
COLLISION	COL SHIFT L
COLOR	COL SHIFT O
CONCAT	C SHIFT O
CONT	nincs
COPY	CO SHIFT P
COS	nincs
DATA	D shift A
DEC	nincs
DCLEAR	DCL SHIFT E
DCLOSE	D SHIFT C
DEF FN	nincs
DELETE	DE SHIFT L
DIM	D SHIFT I
DIRECTORY	DI SHIFT R
DLOAD	D SHIFT L
DO	nincs
DOPEN	D SHIFT O
DRAW	D SHIFT R
DSAVE	D SHIFT S
DVERIFY	D SHIFT V
EL	nincs
END	nincs
ENVELOPE	E SHIFT N
ER	nincs
ERR$	E SHIFT R

Kulcsszó	*Rövidítés*
EXIT	EX SHIFT I
EXP	E SHIFT X
FAST	nincs
FETCH	F SHIFT E
FILTER	F SHIFT I
FOR	F SHIFT O
FRE	F SHIFT R
FNXX	nincs
GET	G SHIFT E
GETKEY	GETK SHIFT E
GET#	nincs
GOSUB	GO SHIFT S
GO64	nincs
GOTO	G SHIFT O
GRAPHIC	G SHIFT R
GSHAPE	G SHIFT S
HEADER	HE SHIFT A
HELP	
HEX$	H SHIFT E
IF...GOTO	nincs
IF...THEN...ELSE	nincs
INPUT	nincs
INPUT#	I SHIFT N
INSTR	IN SHIFT S
INT	nincs
JOY	J SHIFT O
KEY	K SHIFT E
LEFT$	LE SHIFT F
LEN	nincs
LET	L SHIFT E
LIST	L SHIFT I
LOAD	L SHIFT O
LOCATE	LO SHIFT C
LOG	nincs
LOOP	LO SHIFT O
MID$	M SHIFT I
MONITOR	MO SHIFT N
MOVESHAPE	nincs
MOVSPR	M SHIFT O
NEW	nincs
NEXT	N SHIFT E
ON...GOSUB	ON...GO SHIFT S
ON...GOTO	ON...G SHIFT O
OPEN	O SHIFT P
PAINT	P SHIFT A
PEEK	PE SHIFT E
PEN	P SHIFT E
PI	nincs
PLAY	P SHIFT L
POKE	PO SHIFT K
POS	nincs
POT	P SHIFT O
PRINT	?
PRINT#	P SHIFT R
PRINT USING	?US SHIFT I
PUDEF	P SHIFT U
RBUMP	RB SHIFT U
RCLR	R SHIFT C

Kulcsszó	*Rövidítés*
RDOT	R SHIFT D
READ	RE SHIFT A
RECORD	R SHIFT E
REM	nincs
RENAME	RE SHIFT N
RENUMBER	REN SHIFT U
RESTORE	RE SHIFT S
RESUME	RES SHIFT U
RETURN	RE SHIFT T
RGR	R SHIFT G
RIGHT$	R SHIFT I
RLUM	nincs
RND	R SHIFT N
RREG	R SHIFT R
RSPCOLOR	RSP SHIFT C
RSPPOS	R SHIFT S
RSPR	nincs
RSPRITE	RSP SHIFT R
RUN	R SHIFT U
RWINDOW	R SHIFT W
SAVE	S SHIFT A
SCALE	SC SHIFT A
SCNCLR	S SHIFT C
SCRATCH	SC SHIFT R
SGN	S SHIFT G
SIN	S SHIFT I
SLEEP	S SHIFT L
SLOW	nincs
SOUND	S SHIFT O
SPC(	nincs
SPRCOLOR	SPR SHIFT C
SPRDEF	SPR SHIFT D
SPRITE	S SHIFT P
SPRSAV	SPR SHIFT S
SQR	S SHIFT Q
SSHAPE	S SHIFT S
STASH	S SHIFT T
STatus	nincs
STEP	ST SHIFT E
STOP	ST SHIFT O
STR$	ST SHIFT R
SWAP	S SHIFT W
SYS	nincs
TAB(	T SHIFT A
TAN	nincs
TEMPO	T SHIFT E
TI	nincs
TI$	nincs
TO	nincs
TRAP	T SHIFT R
TROFF	TRO SHIFT F
TRON	TR SHIFT O
UNTIL	U SHIFT N
USR	U SHIFT S
VAL	nincs
VERIFY	V SHIFT E
WAIT	W SHIFT A
WHILE	W SHIFT H
WIDTH	WI SHIFT D
XOR	X SHIFT O

L) Függelék

A LEMEZPARANCSOK ÖSSZEFOGLALÁSA

Ez a függelék a Commodore 128-ason a C128-as és a C64-es módban a lemezműveletekkel kapcsolatos parancsokat sorolja fel. Ezekről a parancsokról részletes tájékoztatót az V. fejezet (BASIC 7.0 kislexikon) ad. A lemezmeghajtóhoz adott kezelési utasítás szintén tartalmaz erre vonatkozó információkat.
A BASIC 7.0 új utasításai csak a C128-as módban használhatók. A BASIC 2.0 parancsai mind a C128-as, mind a C64-es módban használhatók.

Parancs	Funkció	BASIC 2.0	BASIC 7.0
APPEND	Adat hozzáfűzése a file-hoz		x
BLOAD	Bináris file betöltése a megadott tárcímtől kezdődően		x
BOOT	Betölt és végrehajt egy programot		x
BSAVE	Kiment egy bináris file-t a megadott tárcímtől kezdődően		x
CATALOG	Kijelzi a képernyőn a lemez tartalomjegyzékét*		x
CLOSE	Lezár egy logikai lemez file-t	x	
CMD	A képernyőre irányuló kivitelt visszaküldi a lemezes filera	x	
COLLECT	Felszabadítja a lemezen egyébként hozzáférésre alkalmatlan helyeket*		x
CONCAT	Összefűz két adatfile-t*		x
COPY	File-okat másol periferikus eszközök között*		x
DCLEAR	A lemezmeghajtón valamennyi nyitott csatornát törli		x
DCLOSE	Lezár egy lemezes logikai file-t		x
DIRECTORY	Kijelzi a képernyőn a lemez tartalomjegyzékét*		x
DLOAD	Betölt egy BASIC programot a lemezről		x
DOPEN	Megnyit egy lemezes file-t olvasásra és/vagy írásra		x
DSAVE	Kiment egy BASIC programot lemezre		x
DVERIFY	Összehasonlítja a tárban levő programot a lemezen levő programmal		x
GET#	Adatbevitel egy megnyitott lemezes fileből	x	
HEADER	Egy lemez formálása*		x
LOAD	Egy file betöltése lemezről	x	
OPEN	Egy file megnyitása ki/bevitelre	x	
PRINT#	Adat kivitele egy fileba	x	
RECORD	Relatív file mutatójának pozicionálása*		x
RENAME	A lemezen levő file nevének megváltoztatása*		x
RUN file-név	Végrehajt egy lemezen levő BASIC programot		x
SAVE	Lemezre ment egy tárban levő programot	x	
VERIFY	Összehasonlítja a tárban és a lemezen levő programot	x	

* Bár ennek a funkciónak nincs pontosan megegyezője a BASIC 2.0-ban, de van erre vonatkozó többcélú parancs. L. erre vonatkozóan a lemezmeghajtóra vonatkozó kezelési útmutató BASIC 2.0 utasításait.

SZAKKIFEJEZÉSEK MAGYARÁZATA

A következő jegyzék a számítástechnikában gyakrabban előforduló szavak és kifejezések rövid meghatározását tartalmazza.

Adat: Számok, betűk vagy jelek, amelyek feldolgozás céljából bekerülnek a számítógépbe.

Adatbázis: A számítógép tárában levő, alkalmas módon szervezett, nagy mennyiségű adat. Az adatbázis-kezelő rendszer olyan program, ami lehetővé teszi az információkhoz való hozzáférést.

Adatráta vagy adatátviteli ráta: Az adatátvitel baudban vagy bit/s-ban (bps) megadott sebessége.

Akusztikus csatoló vagy akusztikus modem: Olyan eszköz, ami a digitális jeleket hanggá alakítja át, telefonvonalakon való továbbítás céljából. Az átviteli sebesség felső határa 1200 baud vagy bit/s (bps) körül van. Vö. Közvetlen csatolású modem.

Alfanumerikus: A billentyűzeten található betűk, számok és speciális jelek, a grafikus karakterek kivételével.

ALU: Aritmetikai logikai egység (Arithmetic Logic Unit). A központi végrehajtó egység (CPU) része, ahol bináris adatokkal folynak a műveletek.

Animáció: A számítógép utasításainak segítségével egy tárgy mozgatása a képernyőn.

Anyalemez (mothorboard): Egy busz-szervezésű rendszerben az a lemez, amelyen a buszvezetékek és azok a csatlakozók vannak, amelyek a rendszer többi lemezének az elhelyezését lehetővé teszik.

ASCII: Az „American Standard Code for Information" rövidítése. Az alfanumerikus karaktereknek egy-egy 7 bitből álló kód felel meg. A kódokkal küldünk információt a billentyűzetről a számítógépbe vagy az egyik számítógépből a másikba. Vö. Karakterfüzér kód.

Assembler: Olyan program, ami az assembly nyelvű utasításokat gépi kódú utasításokra fordítja le.

Assembly nyelv: Géporientált nyelv, amelyben minden egyes gépi kódú utasításnak egy-egy mnemonik felel meg. Mindegyik CPU-nak saját, specifikus assembly nyelve van. Vö. CPU és gépi kód.

Aszinkron átvitel: Adatok szabálytalan időközökben történő továbbítása. A telefonvonalon keresztüli átvitel legnagyobb sebessége 2400 baud. Vö. Szinkron átvitel.

Baud: Soros adatátvitel átviteli sebességének mérőszáma. A táviratküldési technikából kölcsönzött fogalom, 300 baud megközelítőleg 30 byte vagy karakter/s átviteli sebességnek felel meg.

BASIC: A „Beginner's All-purpose Symbolic Instruction Code" (általános célú, szimbolikus utasításkód kezdők számára) rövidítése.

Billentyűzet: Egy számítógépes rendszer beviteli egysége.

Bináris: Kettes alapú számrendszer. Valamennyi számot nullák és egyesek sorozata alkotja.

Bit: A *B*inary dig*IT* rövidítése. A bit a számítógépen belül a legkisebb egység. Egy-egy bit csak két értéket vehet fel, 0-t vagy 1-et. Egy bitet „engedélyezettnek", bekapcsoltnak vagy „magasra állított"-nak nevezünk akkor, ha az értéke 1, és „letiltott"-nak, kikapcsoltnak vagy „alacsonyra állított"-nak akkor, ha az értéke 0.

Bittérképes üzemmód: A Commodore 128-as grafikus üzemmódja, amelyben lehetőség van arra, hogy a képernyő minden egyes képernyőpontját külön-külön vezéreljük.

Bitvezérlés: Olyan soros adatátviteli mód, amelynél mindegyik bit szignifikáns, és egyetlen karaktert egy start és egyetlen karaktert egy start és egy stop bit fog közre.

Buborékmemória: A számítógép viszonylag új típusú tára, amely kicsiny mágneses „buborékokat" használ az adatok tárolására.

Burkológörbe generátor: A Commodore 128-as része, amely a zenei hangok számára bizonyos hullámformákat állít elő (fűrészfog, háromszög, impulzus és zaj). Vö. Hullámforma.

Busz: Párhuzamos vagy soros vonalak, amelyeken az egyes készülékek közötti jelforgalom bonyolódik le. A számítógépeket gyakran a buszszerkezetükkel jellemzik.

Buszvonalak hálózata: Olyan rendszer, amelyben a számítógép egyes egységei egy közös elosztóbusz vagy -csatorna segítségével egymás között kapcsolatot tartanak.

Byte: 8 bitből álló sorozat, ami a számítógépben a címezhető tár legkisebb egysége. A Commodore 128-as mindegyik tárcímén 1 byte-nyi információ helyezhető el. A tárban 1 karakter tárolásához 1 byte-nyi táregységre van szükség.

Chip: Miniatűr elektronikus áramkör, ami a számítógépben a grafikával, a hanggal, a bevitellel/kivitellel kapcsolatos műveleteket végzi el.

Ciklus: Egy olyan programrész, amelyben műveletek sorozata meghatározott alkalommal ismétlődik.

Cím: Az a címke vagy szám, ami a regiszter vagy a tár azon helyének az azonosítására szolgál, ahol az információ található.

Compiler: Fordítóprogram, amely egy magas szintű nyelven, pl. BASIC nyelven megírt programot gépi kódúra fordít.

CPU: A „Central Processing Unit" (központi végrehajtó egység) rövidítése.

Datasette: Kazettás magnetofonkészülék, ami programokat és adatfile-okat tárol szalagon, soros formában.

Dekrementálás: Egy indexváltozó vagy számláló bizonyos értékkel való csökkentése.

Digitális: Számítógépek és adatátviteli eljárások technológiája; valamennyi információbitek sorozataként van kódolva. Egy bit értéke lehet 1, ez jelenti a bekapcsolt állapotát; és lehet 0, ez jelenti a kikapcsolt állapotát.

Dimenzió: Egy tömböt jellemez. Megadja, hogy a tömb elemei hány tengely irányában kerülnek tárolásra. Így pl. egy kétdimenziós tömbnek van egy X-tengelye a sorok és egy Y-tengelye az oszlopok jelölésére. Vö. Tömb.

Direkt mód: Ebben az üzemmódban a BASIC parancsok a RETURN gomb megnyomása után azonnal végrehajtódnak. Parancs módnak is nevezik. Vö. Parancs.

Duplex üzemmód: Lehetővé teszi, hogy két számítógép egy időben küldjön és fogadjon adatokat.

Egész szám: Olyan szám, amely nem tartalmaz törtrészt; pl. 0, 1, 2 stb.

Elágazás: Ugrás egy programrészre, és annak végrehajtása. Ilyen BASIC ugróutasítás pl. a GOTO és a GOSUB.

Elektronikus postaszolgálat: Számítógépet használók részére nyújtott kommunikációs szolgáltatás. A szöveges üzenetek egy központi számítógépbe, az ún. elektronikus „postaládába" kerülnek, ahonnan a címzett ezeket később lehívhatja.

Elengedési idő (Release): Az az időtartam, ami alatt a zenei hang hangereje a kitartási szintről (sustain) nullára csökken.

Engedélyezés: Egy bit, byte vagy a számítógép valamely műveletének bekapcsolása.

EPROM: Olyan PROM egység, amelyet a használó is törölhet, általában ibolyántúli fény segítségével. Vö. PROM.

Értékadó utasítás: Olyan BASIC utasítás, ami egy változót, állandót vagy tömbelemet egy bizonyos szám- vagy füzérértékkel tesz egyenlővé.

Felbontás: A képernyőn a pixelek (képpontok) sűrűsége, ami az ábrázolt kép részletgazdagságát határozza meg.

Félduplex üzemmód: Az adatok egy időben csak egyirányú átvitelét teszi lehetővé; ha az egyik készülék adatokat ad, akkor a másik csak fogadhatja ezeket.

Felfutási idő (attack): Hangkeltésnél az az idő, mialatt a hangerő a nulláról a maximális értékre fut fel.

Feltétel: Az IF és THEN szavak között levő kifejezés(ek), amely(ek) kiértékeli(k), hogy az IF... THEN utasításpárban megadott feltétel teljesül-e vagy nem. Az IF...THEN utasításpár képessé teszi a számítógépet arra, hogy döntéseket hozzon.

File: Egy egységként tekintett program vagy adatgyűjtemény, amely lemezen vagy szalagon van tárolva.

Firmware: A számítógép ROM-ban tárolt utasításai (egy játékcartridge-on tárolt programhoz hasonló utasítások).

Forráskód: Magas szintű nyelven megírt, nem végrehajtható program. Egy compilernek (fordí-

tóprogramnak) vagy egy assemblernek le kell fordítania a programot objekt kódba (gépi kódba), hogy azt a számítógép megértse.

Frekvencia: A hang által keltett hullámok másodpercenkénti száma. A frekvencia a hallható hangok magasságával változik.

Funkcióbillentyűk: A Commodore 128-as jobb oldalán levő négy billentyű. Mindegyik billentyű programozható utasítássorozatok végrehajtására. A billentyűk a SHIFT gombbal együtt is használhatók, így nyolc különböző utasítássorozat programozható.

Függvény: Olyan, előre meghatározott művelet, aminek az eredménye egyetlen érték.

Füzér: Idézőjelek között levő karakterek vagy alfanumerikus karakterek sorozata.

GCR formátum: Információk lemezen való tárolásának formátuma. A 1541-es és a 1571-es lemezmeghajtóval használva a GCR formátumú lemezek írhatók és olvashatók.

Gépi kód: A legalacsonyabb szintű nyelv, amelyet a számítógép megért. A számítógép valamennyi magas szintű nyelvet, mint pl. a BASIC nyelvet is gépi kódra fordít le, mielőtt bármely utasítását végrehajtaná. A gépi kód bináris számrendszerben van írva, amelyet a számítógép is megért. Az ilyen módon megírt programot gépi kódú programnak vagy tárgykódnak is nevezik.

Grafika: A számítógép tárában levő adatokkal előállított és a képernyőn megjelenő ábrák (karakterek, szimbólumok és képek).

Grafikus karakterek: A számítógép billentyűzetén levő, nem alfanumerikus karakterek.

Gyűrűs hálózat: Olyan rendszer, amelynél valamennyi állomás úgy van kapcsolva, hogy ezek egy folyamatos hurkot alkotnak.

Hardver: Egy számítógéprendszerben fizikailag is meglevő eszközök; ilyenek a billentyűzet, a lemezmeghajtó egység vagy a nyomtató.

Háttér színe: A képernyő azon részének a színe, ahol a karakterek megjelennek.

Helyi hálózat (Local Network): A kistávolságú adatátviteli eljárások különböző fajtájának egyike. Lehetővé teszi, hogy egy közös átvivőeszközt több készülék nagy átviteli sebességgel használhasson. Elnevezéseként a „Local Area Network" (LAN) is használatos.

Hexadecimális: 16-alapú számrendszer. A gépi nyelvű programokat többnyire hexadecimális számrendszerben írják.

Hibavizsgálat vagy hibaészlelés: Szoftver rutin, ami a hibás adatot azonosítja, és gyakran javítja is.

Home: A képernyő bal felső sarkát jelöli.

Hullámforma: A hanghullám alakjának grafikus ábrázolása. A hullámforma a hang néhány fizikai jellemzőjét határozza meg.

IC (Integrated Circuit): Integrált áramkör. Egy szilíciumszelet (chip), amely tranzisztorokból, diódákból, ellenállásokból és kondenzátorokból álló áramköröket tartalmaz. Az integrált áramkörök a korábbi számítógépekben használt egyedi áramkörökhöz képest kisebbek, gyorsabbak és hatékonyabbak.

Időzítés: Eljárás egy adatokat küldő és fogadó egység közötti adatforgalom szinkronizálására, amelyek a bináris információ kódolása céljából modulálva vannak.

Index: Egy FOR...NEXT ciklusban a változó értékét számlálja.

Inkrementálás: Egy indexváltozó vagy egy számláló bizonyos értékkel való növelése.

I/O: Input/Output (bevitel/kivitel). Adatoknak a számítógépbe való bevitelét, ill. a számítógépből lemezre, nyomtatóra vagy valamilyen tárolóeszközre való kivitelét jelenti.

Input: Bevitel, amely révén a számítógépbe adatok kerülnek feldolgozás céljából. Beviteli forrás lehet a billentyűzet, a lemezmeghajtó egység, a Datassette kazettás magnetofon vagy a modem.

Interface: Csatlakozó felület a számítógép és a „külvilág" (operátor, periferikus egység vagy bármely kommunikációs egység) között. A számítógéphez illeszkedő egység lehet fizikai, mint pl. egy periféria vagy lehet logikai, ami tulajdonképpen egy szoftver.

Kapu (port): Csatorna, amelyen keresztül adatok kerülnek a CPU-ba, ill. amelyen keresztül a CPU adatokat küld. Egy 8 bites CPU 256 kaput tud megcímezni.

Karakter: A számítógép billentyűzetén levő mindaz a jel, ami a képernyőn megjeleníthető. Karakterek a számok, a betűk, az írásjelek és a grafikus szimbólumok.

Karakterfüzér kód: Az a számérték, ami a számítógép tárában a Commodore 128-as valamelyik karakteréhez van hozzárendelve.

Karakterkészlet: Egymáshoz tartozó karakterek csoportja. A Commodore 128-as karakterkészletei a nagybetűk, a kisbetűk és a grafikus karakterek készlete.

Karaktertár: A Commodore 128-as tárának az a része ami a képernyőn megjelenő karakterek kódolt mintáit tárolja.

Képernyő: Video kijelzőegység, ami lehet televíziós vevőkészülék vagy videomonitor.

Képernyőkód: Az a szám, amely egy, a képernyőtárban levő karakternek felel meg. Ha a billentyűzeten lenyomunk egy karaktert, akkor ennek a karakternek a képernyőkódja automatikusan beíródik a képernyőtárba. Egy karaktert úgy is megjeleníthetünk, hogy a képernyőkódját a POKE utasítással közvetlenül beírjuk a képernyőtárba.

Képernyőtár: A Commodore 128-as tárának az a területe, amely a képernyőn kijelzésre kerülő információt tartalmazza.

Keret színe: A képernyőt körülvevő keret színe.

Késleltető ciklus: Egy üres FOR...NEXT ciklus, amellyel a program végrehajtását lassítani lehet.

Kifejezés: Logikai, matematikai vagy relációs műveleti jel, ami állandók, változók vagy tömbelemek kombinációját adja meg, és a végeredménye egy számérték.

Kilobyte (K): 1024 byte.

Kitartási szint (sustain): Egy zenei hang kitartott hangereje.

Koaxiális kábel: Átviteli eszköz, amit általában helyi hálózatoknál alkalmaznak.

Kompozit monitor: A 40 oszlopos kép megjelenítését lehetővé tevő eszköz.

Koordináta: Vízszintes és függőleges vonalakból álló háló egyetlen pontja, amelynek vízszintes (X) és függőleges (Y) irányban értéke van.

Közvetlen csatolású modem (Direct Connect Modem): Olyan eszköz, amely egy számítógép digitális jeleit elektronikus impulzusokká alakítja át, hogy azokat telefonvonalakon továbbítani lehessen. Az akusztikus csatoló ellentettje.

Kurzor: Villogó négyzet, ami a képernyőn azt a helyet jelöli, ahova a következő karakter kerül.

Lecsengési idő (decay): Az az időtartam, mialatt a hangerő a maximális értékéről egy közbenső szintre – az ún. kitartási szintre (sustain) – csökken. Vö. Kitartási szint.

Lemezmeghajtó: Nagyméretű file-ok, ill. programok hajlékony lemezen való tárolására, ill. arról való betöltésére szolgáló véletlen (random) elérésű eszköz.

Lemézoperációs rendszer (Disk Operating System): Az információnak a lemezről, ill. a lemezre való átvitelét végző program. Közismert megnevezése a DOS.

Letiltás: Egy bitnek, byte-nak vagy a számítógép valamely műveletének kikapcsolása.

Mátrix: Kétdimenziós tömb, ahol a soroknak és oszlopoknak értékeik vannak.

Megjegyzés (remark): A program dokumentálására használatos. Azt, ami a megjegyzésekben szerepel, a számítógép nem hajtja végre, de a listázáskor kijelzi.

MFM: Eljárás információk lemezen való tárolására. A 1541-es és a 1571-es meghajtóegységek olvashatják, de nem írhatják felül az ilyen formátumú lemezeket.

Mikroprocesszor: Egy CPU, amelyet egyetlen integrált áramkör (IC) alkot. A Commodore személyi számítógépek 6510-es, 6502-es és Z80-as mikroprocesszorokat használnak.

Modem: A *MO*dulátor/*DEM*odulátor rövidítése. Olyan eszköz, amely a számítógép digitális jeleit olyan elektromos impulzusokká alakítja át, amelyeket telefonvonalakon keresztül továbbítani lehet, és ugyanez fordítva.

Mód: A működés valamely fajtája.

Monitor: A televíziós vevőkészülékhez hasonlítható képmegjelenítő eszköz, de ahhoz képest nagyobb a felbontóképessége (élesebb képet ad).

Mutató: Regiszter, amely egy tárcímet tartalmaz (= a tárcímre mutat).

Műveletek sorrendje: Az a sorrend, amely szerint a számítógép a műveleteket végrehajtja. A műveletek hierarchiájának is nevezik.

Műveleti jel: Olyan szimbólum, amely megmondja a számítógépnek, hogy a kifejezésekben megadott változókkal, állandókkal vagy tömbelemekkel matematikai, logikai vagy relációs műveletet kell-e végrehajtani. Matematikai műveleti jelek: +, −, * és ↑. Relációs műveleti jelek: <, =, >, < =, > =, < >. Logikai műveleti jelek: AND, OR, NOT és XOR (ÉS, VAGY, NEM, kizáró VAGY).

Nulla füzér: Üres karakter (""). Olyan karakter, amelyhez még nincs megadva karakterfüzér kód. GET utasításban használva az *Illegal quantity error* hibajelzést kapjuk.

Nyomtató: Olyan periferikus egység, amely a számítógép tárának tartalmát papírra nyomtatja. Az így kinyomtatott papírt hard-copynak is nevezik.

Operációs rendszer: Beépített program, amely a számítógép működését vezérli.

Óra: A mikroprocesszor működésének időzítését végző áramkör.

Parancs: Direkt (vagy parancs) módban használt BASIC utasítás valamely feladat elvégzésére. Vö. Direkt mód.

Paritásbit: Egy „1" vagy egy „0", amely bitek egy csoportjához adódik hozzá, és a bitek összegét aszerint azonosítja, hogy az páros vagy páratlan.

Párhuzamos kapu (port): Olyan bemeneti/kimeneti kapu, amely lehetővé teszi, hogy többeres huzalokon keresztül egy időben egy byte-ot lehessen továbbítani.

Periféria: A számítógéphez kapcsolt külső egység, mint pl. lemezmeghajtó, nyomtató, modem vagy botkormány.

Piexel: A képpont számítógépes elnevezése. A képernyőn bármely megjeleníthető pont egy pixel. A képernyőn mindegyik karaktert pixelek 8*8-as pontrácsa jelenít meg. A teljes képernyőt egy 320*200 pixelből álló rács alkotja. A bittérképes üzemmódban mindegyik pixel a számítógép tárában levő egy bitnek felel meg.

Polling: Többterminálos számítógépes rendszerben a kommunikáció vezérlésének egyik módja. Egy „mester" (master) állomás a közös adatátviteli rendszerre csatlakozó többi állomást rendszeresen lekérdezi arra vonatkozóan, hogy van-e küldeni való információjuk.

Program: Utasítások sorozata, amely megmondja a számítógépnek, hogy milyen feladatot végezzen el. A programok tárolhatók lemezen vagy szalagon, lehetnek a számítógép tárában, és kiírhatók a nyomtatóra.

Programozható: Alkalmas arra, hogy számítógépes utasításokkal feldolgozható legyen.

Programsor: Egy programban egy sorszám és az ezt követő utasítás vagy utasítások sorozata. A Commodore 128-asnál egy programsor maximum 160 karaktert tartalmazhat.

PROM: A „*P*rogrammable *R*ead *O*nly *M*emory" (programozható ROM) rövidítése. Olyan félvezetős tár, amelynek a tartalmát nem lehet megváltoztatni.

Protokoll: Azok a szabályok, amelyek szerint a számítógépek egymás között kicserélik az információt, beleértve a továbbítandó adategységek szervezését is.

RAM (*R*andom *A*ccess *M*emory): A számítógép tárának az a területe, amit a programozó szabadon használhat, ahová írni, ill. ahonnan olvasni lehet. Valamennyi RAM tárcím egyformán elérhető bármikor és bármilyen sorrendben. A számítógép kikapcsolásakor a RAM tartalma törlődik.

Rács: Sorokból és oszlopokból álló, kétdimenziós mátrix. A rácsot általában a képernyőn mozgatható alakzatok (sprite-ok) és programozható karakterek tervezésére használjuk.

Regiszter: Valamely tárcím a RAM-ban. Mindegyik regiszter egy byte-ot tartalmaz. Egy regiszter egy 0...255 közé eső bináris értéket tartalmazhat.

RGBI monitor: *R*ed/*G*reen/*B*lue/*I*ntensity = vörös-zöld-kék-intenzitás). Nagy felbontóképességű megjelenítő egység, ami a 80 oszlopos képernyőformátum megjelenítéséhez szükséges.

ROM (*R*ead *O*nly *M*emory): A számítógép tárának állandó tartalmú része. A ROM tárcímek tartalma olvasható, de nem írható felül. A Commodore 128-as ROM-ja tartalmazza a BASIC értelmezőt (interpreter), a karaktergenerátort és az operációs rendszer egy részét.

RS–232: A soros átviteli kapuk elektromos és mechanikai specifikálásra ajánlott szabvány. A Commodore 128-as párhuzamos felhasználói kapuját (parallel user port) szoftver útján soros kapuként (serial port) is lehet kezelni, esetleg egy csatoló (interface) közbeiktatásával.

SID (*S*ound *I*nterface *D*evice): A MOS 6581-es chip a Commodore 128-as szintetizátora. L. erre vonatkozóan a Commodore 128-as programozói kézikönyvét.

Soros kapu: Adatok soros átvitelére szolgáló kapu; a bitek átvitele egyetlen huzalon, egymást követően történik.

Sprite: Programozható, mozgatható, nagy felbontású grafikus ábra. MOB-nak (*M*ovable *O*bject *B*lock) is nevezik.

Standard karakter mód: A Commodore 128-asnak az az üzemmódja, amelybe bekapcsoláskor kerül, és amikor a programokat írjuk.

Start bit: Egy bit vagy bitek csoportja, amely egy adatszó kezdetét azonosítja.

Stop bit: Egy bit vagy bitek csoportja, amely egy adatszó végét azonosítja, és két adatszó között meghatározza a szóközt.

Szalagkábel: Egymással párhuzamosan vezetett elektromos huzalok csoportja.

Számítógép: Elektronikus, digitális készülék, amely információt tárol és dolgoz fel.

Számláló: Változó, ami figyelemmel kíséri, hogy egy programban egy esemény hányszor fordult elő.

Szinkron átvitel: A vevő és az adó egység között szinkronizáló vagy órajelet használó adatátviteli módszer.

Szintaxis: A programozói nyelv nyelvtani szabályai.

Színtár: A Commodore 128-as tárának az a része, amely a képernyőtár minden egyes címén levő színt vezérli.

Szó: Azoknak a biteknek a száma, amelyet a CPU mint egy egységet kezel. Egy 8 bites gépben a szóhosszúság 8 bit; egy 16 bites gépben a szóhosszúság 16 bit.

Szoftver: Számítógép programok (utasítások sora), amelyek lemezen, szalagon vagy cartridgeon vannak tárolva, és amelyek betölthetők a számítógép tárába. A szoftver lényegében az, ami megmondja a számítógépnek, hogy mit végezzen el.

Szólam: A SID chipen belüli hangkeltő komponens. A SID chip segítségével a Commodore 128-as egy időben három különböző hangot tud előállítani. Mindegyik szólam tartalmaz egy hangoszcillátort, egy hullámforma-generátort, egy burkológörbe-generátort és egy amplitúdómodulátort.

Szubrutin: Egy külön feladatot ellátó, önmagában is működőképes, a főprogramtól elválasztható programrészlet. A főprogramból a szubrutinok hívása a GOSUB utasítással történik és a szubrutinnak a RETURN utasítással kell végződnie.

Tár (memória): A számítógépen belül a tárolásra alkalmas hely. A tár két különböző típusa a ROM és a RAM típusú tár.

Tárcsázható vonal: Normál telefonvonal, amelyet adatok továbbítására lehet használni.

Tárhely: A számítógépen belül egy tárcímmel meghatározott hely. A Commodore 128-asnak 131 072 (0–131 071) tárhelye van.

Többmunkahelyes hálózat: Olyan rugalmas rendszer, amelyben mindegyik munkahely bármikor hozzáférhet a hálózathoz. A rendszer arról is gondoskodik, hogy akkor se lépjen fel zavar a működésben, ha két számítógép ugyanazon időben akarna adatokat továbbítani.

Többszínű bittérképes üzemmód: Olyan grafikus üzemmód, ami lehetővé teszi, hogy egy 8*8-as pontrácson belül minden egyes pixelt négy szín valamelyikével jelenítsünk meg. Vö. Pixel.

Többszínű karakter-üzemmód: Olyan grafikus üzemmód, ami lehetővé teszi, hogy egy 8*8-as pontrácsú karakter négyszínű legyen.

Tömb: Adattárolási szerkezet, amelyben egymáshoz tartozó állandók vagy változók sora egymás utáni tárcímeken található. A tömbben levő állandót vagy változót a tömb egy elemének nevezzük. Az elemhez az elem tömbön belüli kijelölésével lehet hozzáférni. Vö. Kijelölés.

Tömörítés (crunch): A számítógép tárában egy program tárolására szolgáló tárterület minimalizálása.

Transzparens: A számítógép olyan műveletét írja le, aminek a végrehajtásához a használónak nem kell beavatkoznia.

Utasítás: Egy programsorban levő BASIC instrukció.

Ütközés észlelése: Többgépes rendszerben elvégzendő feladat, ami megakadályozza, hogy két számítógép egymás munkáját zavarja.

Változó: Egy tárrész, ami egy változó füzért vagy számértéket jelöl. A változó nevei tetszőleges hosszúságúak lehetnek, de a Commodore 128-as csak az első két karaktert tárolja. Az első karakternek betűnek kell lennie.

Véletlenszám: Egy 9 helyiértékű, decimális szám 0.000000001 és 0.999999999 között, amelyet a *RaND*om (RND) függvény állít elő.

VIC (*V*ideo *I*nterface *C*ontroller): A MOS 6566-os chip határozza meg a Commodore 128-as 40 oszlopos üzemmódú grafikai jellemzőit. L. ehhez a Commodore 128-as programozói kézikönyvét.

Vivőfrekvencia: Az egymással kapcsolatban álló egységek közötti konstans jel, ami a bináris információ kódolása céljából modulálva van.

JEGYZET

Ára